Intro to ML - Exam 2

Shivarjun Sarkar, Yash Warty, Sahil Natu and Vivek Mehendiratta

16/08/2021

Github: https://github.com/ShivarjunSarkar/Intro-to-ML-Exam_2/

Problem 1: Green Buildings

First we will go ahead and setup the required libraries:

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.2     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

## Registered S3 method overwritten by 'mosaic':
##   method                           from
##   fortify.SpatialPolygonsDataFrame ggplot2

##
## The 'mosaic' package masks several functions from core packages in order to add
## additional features.  The original behavior of these functions should not be affected by this.

##
## Attaching package: 'mosaic'

## The following object is masked from 'package:Matrix':
##
##     mean

## The following objects are masked from 'package:dplyr':
##
##     count, do, tally

## The following object is masked from 'package:purrr':
##
##     cross
```

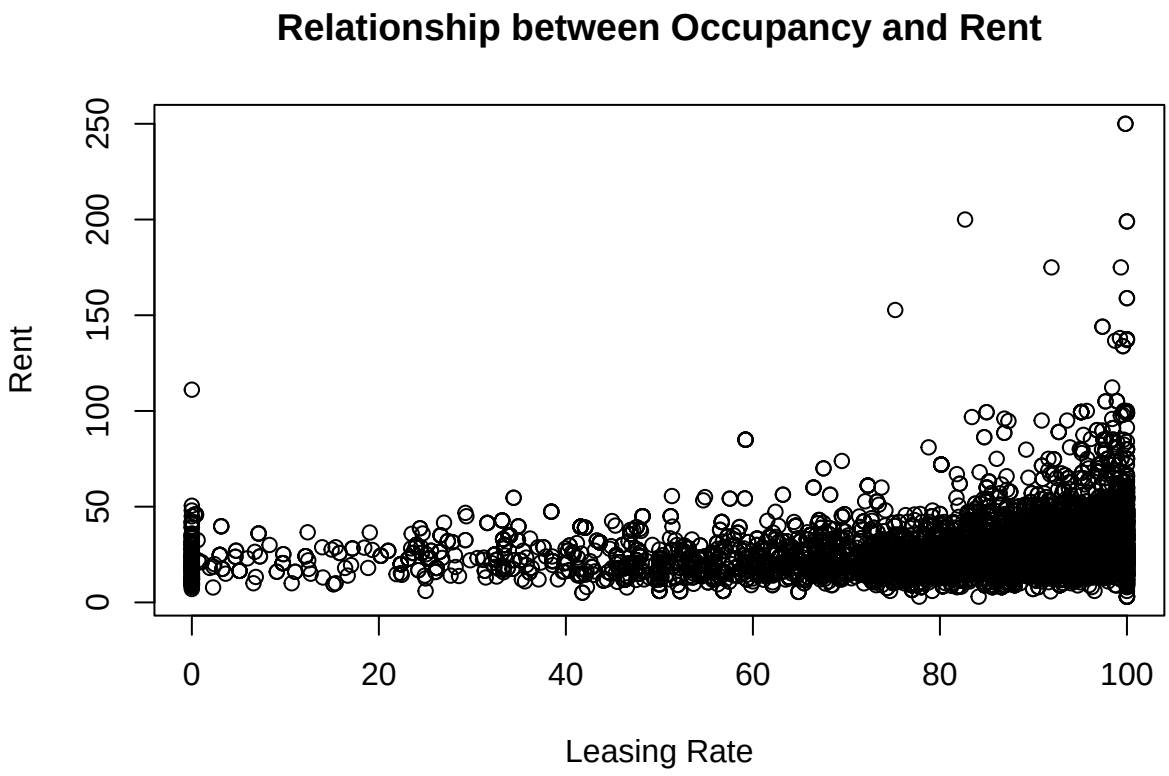
```
## The following object is masked from 'package:ggplot2':
##
##     stat

## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cor.test, cov, fivenum, IQR, median, prop.test,
##     quantile, sd, t.test, var

## The following objects are masked from 'package:base':
##
##     max, mean, min, prod, range, sample, sum
```

Loading in the Green Buildings data

Dropping the NA values from the empl_gr column for use in later calculations:

Point 1

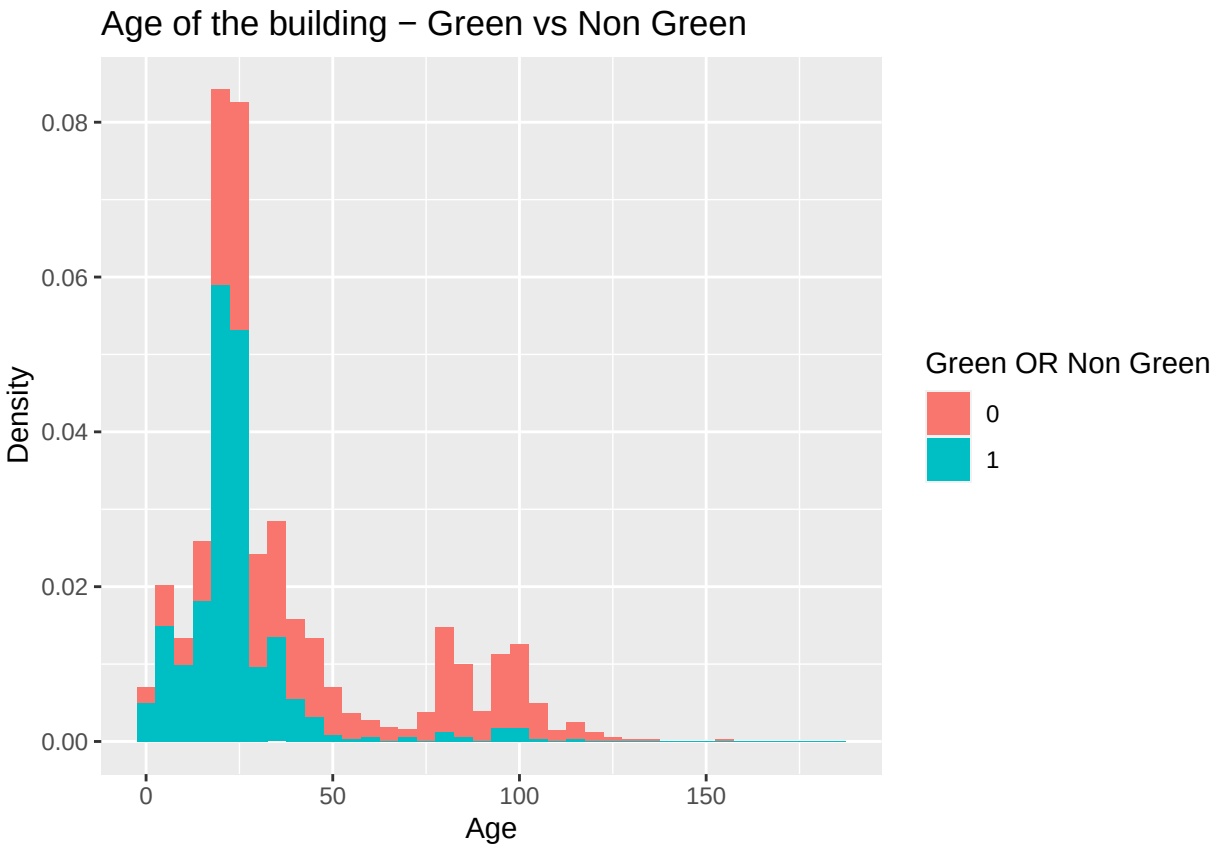
Checking the analyst claim of low leasing rate having weird stuff going on



Looking at the simple plot, it seems like there is a positive correlation. There is not anything weird in the data and so we can use all entries.

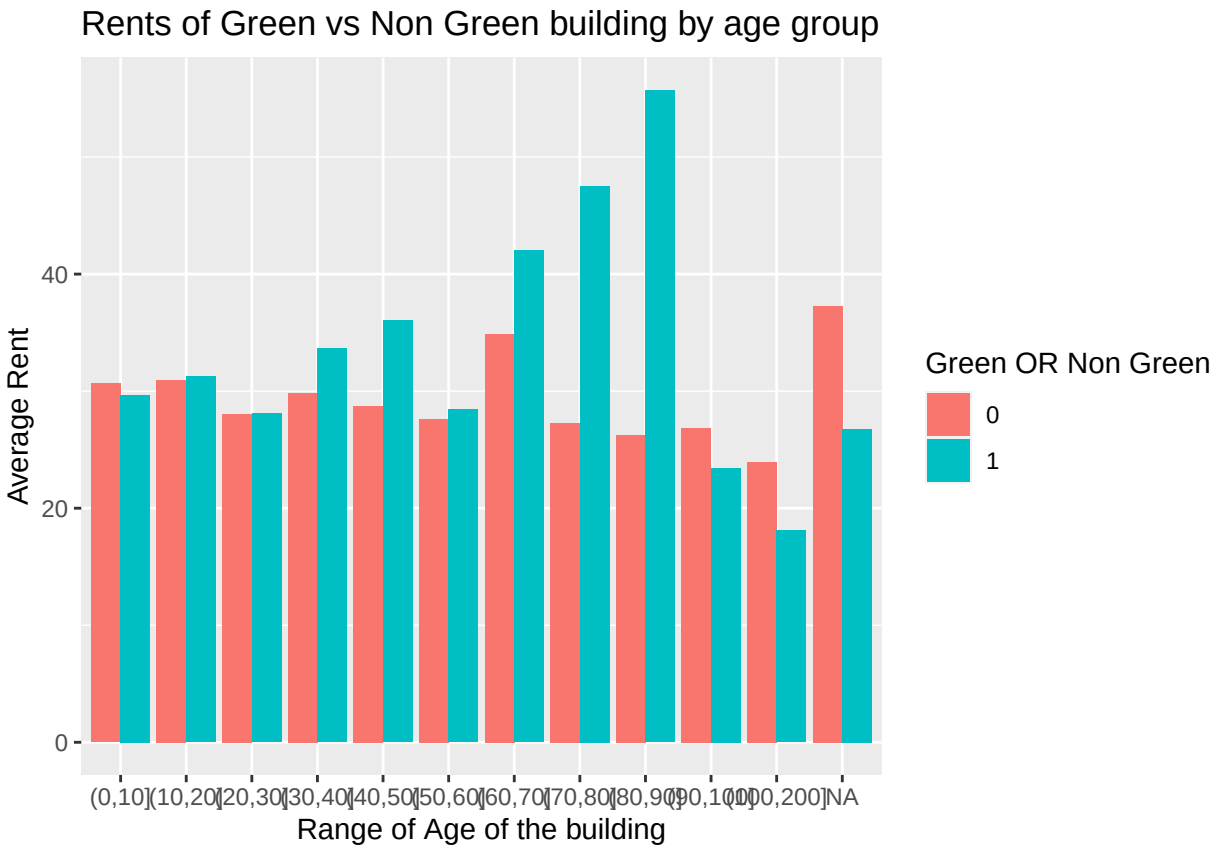
Point 2

Logically, newer buildings should have higher rents, even if they are not green. The analyst has not really considered this factor. Let us understand the distribution of buildings in our data:



We can see that most buildings older than 50 years are non-green, since green ratings are a new phenomenon. Building on this point, checking for differences in rent between Green and Non-Green across age groups:

```
## `summarise()` has grouped output by 'binnedage'. You can override using the `.groups` argument.
```

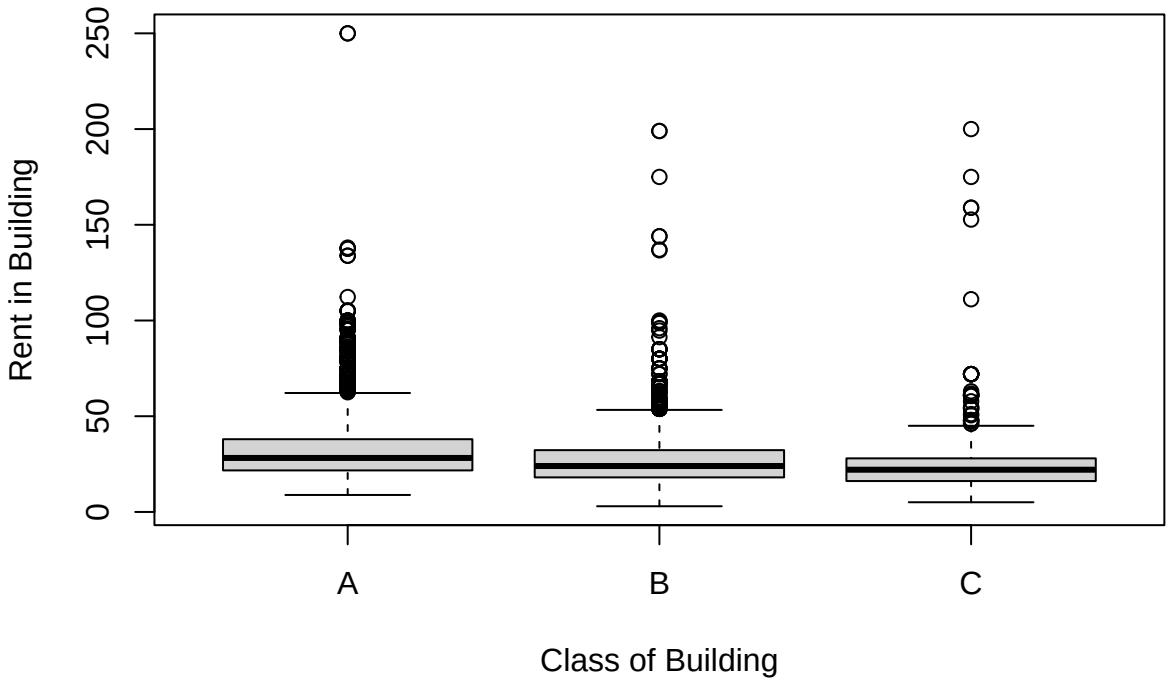


The premium for non-green buildings is higher in the initial years, with green buildings only commanding significantly higher rents after about 30 years. So the analyst calculation of recouping costs may be incorrect.

Point 3

Class A buildings should ideally command the highest rent, with Class C having the lowest rents.Creating the Classifications by Class

Grouping by Class



From the above plot, we see that buildings in Class A do have higher rents than those in Class B and C. Let us now check for differences in rent between Green and Non Green buildings:

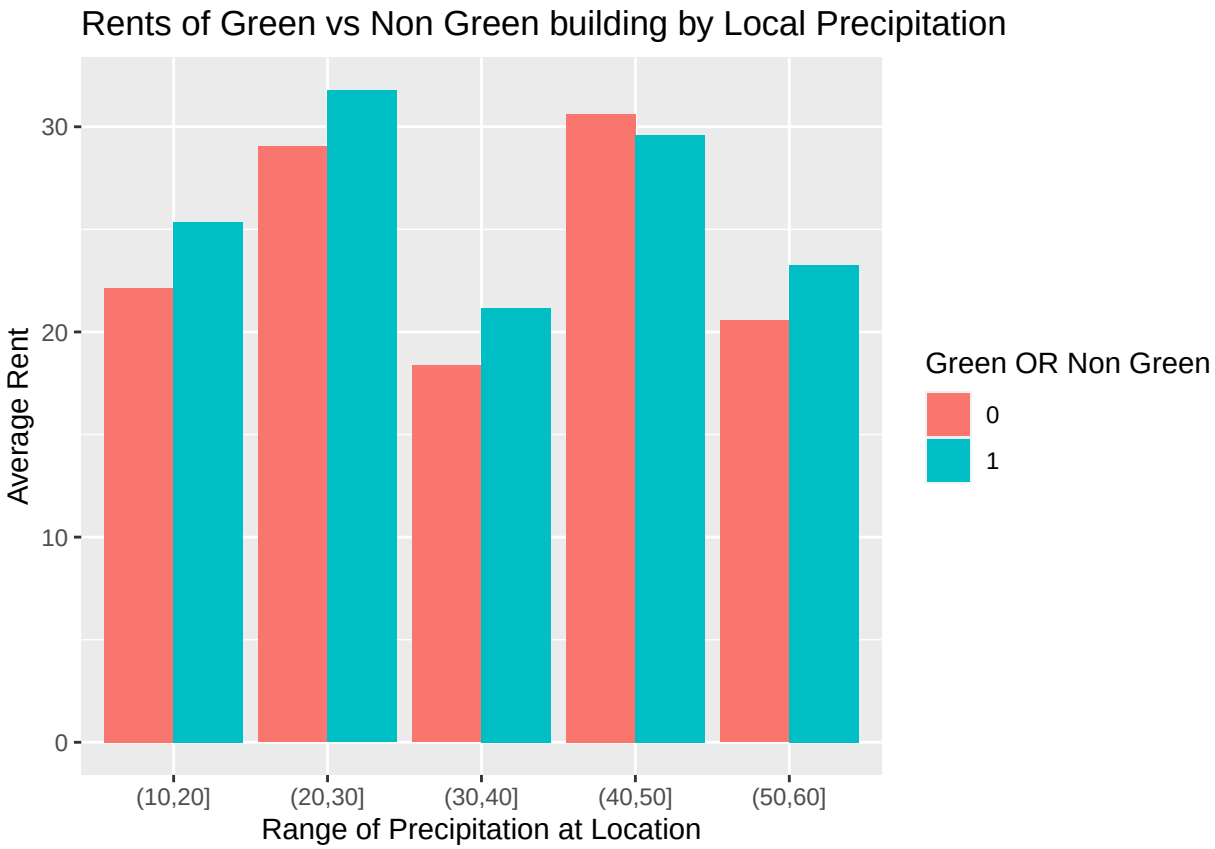
```
## `summarise()` has grouped output by 'classes'. You can override using the `.groups` argument.
```

As we can see, Class A and B (better quality buildings) have higher rents for Non-Green buildings and so there is no benefit to be derived from a Green building.

Point 4

Austin does get a lot of rain in the summer, so maybe tenants prefer green buildings since these are built to the LEED code and will have additional facilities like rain-water harvesting and lightning resistors, that may keep them safe and prevent damage.

```
## `summarise()` has grouped output by 'binnedprec'. You can override using the `.groups` argument.
```

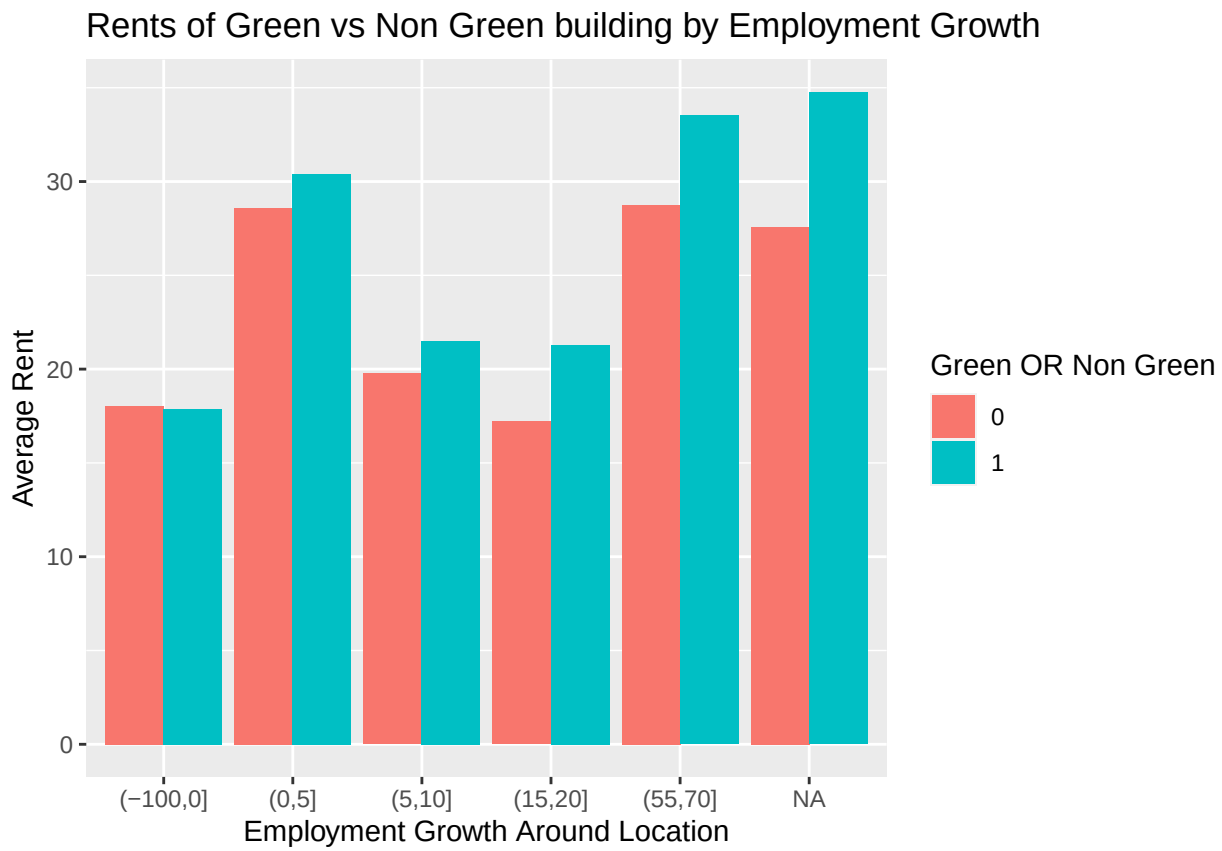


Austin gets about 33-36 inches of rain every year. If we look at buildings in that range, it does appear that green buildings have a slight advantage over non-green counterparts. However, there does not appear to be a significant difference to warrant investing the additional money and taking longer to get returns on our investment.

Point 5

Austin is a booming city especially for younger populations. So employment growth may be a larger factor in rent determination rather than green or non-green. The analyst did not consider this:

```
## `summarise()` has grouped output by 'binnedempl'. You can override using the `.groups` argument.
```



Employer growth appears to prove the analysts point of Green buildings having higher rents, especially in a booming city like Austin.

Conclusion

After checking for various parameters, the cons of investing extra money in a green building, appear to outweigh the pros. So overall, there is no strong case for developing green buildings yet.

Problem 2: Flights at ABIA

Visual story telling part 2: flights at ABIA

The flight data contains data for all flights that departed from and arrived at the Austin airport. We will analyze this data to search for patterns and answer some questions.

```
##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths

## [1] 99260    29
```

```
##  [1] "Year"              "Month"             "DayofMonth"
##  [4] "DayOfWeek"         "DepTime"           "CRSDepTime"
##  [7] "ArrTime"           "CRSArrTime"        "UniqueCarrier"
## [10] "FlightNum"         "TailNum"           "ActualElapsedTime"
## [13] "CRSElapsedTime"    "AirTime"           "ArrDelay"
## [16] "DepDelay"          "Origin"            "Dest"
## [19] "Distance"          "TaxiIn"            "TaxiOut"
## [22] "Cancelled"         "CancellationCode"  "Diverted"
## [25] "CarrierDelay"      "WeatherDelay"      "NASDelay"
## [28] "SecurityDelay"     "LateAircraftDelay"
```

The data consists of 99260 rows and 29 variables. The variable names have been listed above.

```
##              Year             Month        DayofMonth         DayOfWeek
##         "integer"         "integer"         "integer"         "integer"
##           DepTime        CRSDepTime           ArrTime        CRSArrTime
##         "integer"         "integer"         "integer"         "integer"
##     UniqueCarrier         FlightNum           TailNum ActualElapsedTime
##       "character"         "integer"       "character"         "integer"
##    CRSElapsedTime           AirTime          ArrDelay          DepDelay
##         "integer"         "integer"         "integer"         "integer"
##            Origin              Dest          Distance            TaxiIn
##       "character"       "character"         "integer"         "integer"
##           TaxiOut         Cancelled  CancellationCode          Diverted
##         "integer"         "integer"       "character"         "integer"
##      CarrierDelay      WeatherDelay          NASDelay     SecurityDelay
##         "integer"         "integer"         "integer"         "integer"
## LateAircraftDelay
##         "integer"
```

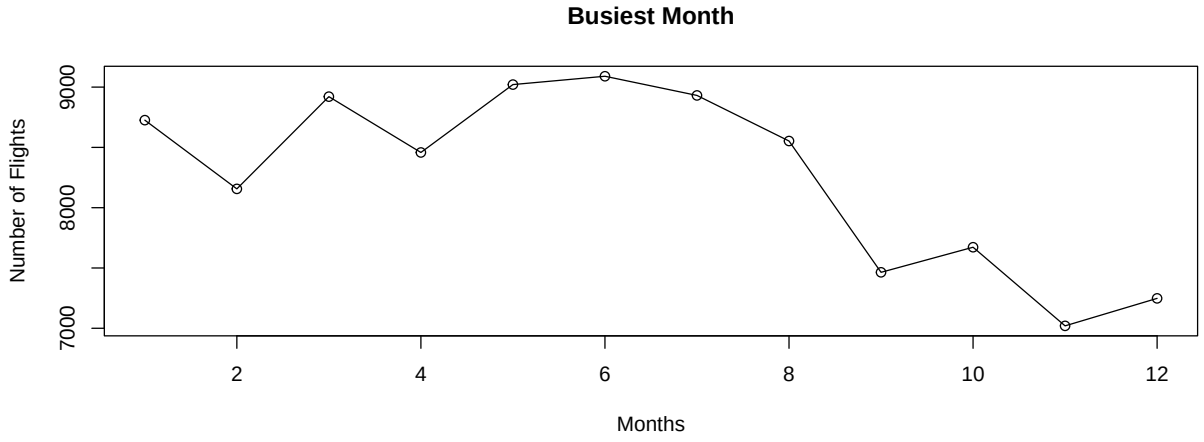
```
##   Year Month DayofMonth DayOfWeek DepTime CRSDepTime ArrTime CRSArrTime
## 1 2008     1          1         2     120       1935     309       2130
## 2 2008     1          1         2     555        600     826        835
## 3 2008     1          1         2     600        600     728        729
## 4 2008     1          1         2     601        605     727        750
## 5 2008     1          1         2     601        600     654        700
## 6 2008     1          1         2     636        645     934        932
##   UniqueCarrier FlightNum TailNum ActualElapsedTime CRSElapsedTime AirTime
## 1            9E      5746  84129E                109            115      88
## 2            AA      1614  N438AA                151            155     133
## 3            YV      2883  N922FJ                148            149     125
## 4            9E      5743  89189E                 86            105      70
## 5            AA      1157  N4XAAA                 53             60      38
## 6            NW      1674   N967N                178            167     145
##   ArrDelay DepDelay Origin Dest Distance TaxiIn TaxiOut Cancelled
## 1      339      345    MEM  AUS      559      3      18         0
## 2       -9       -5    AUS  ORD      978      7      11         0
## 3       -1        0    AUS  PHX      872      7      16         0
## 4      -23       -4    AUS  MEM      559      4      12         0
## 5       -6        1    AUS  DFW      190      5      10         0
## 6        2       -9    AUS  MSP     1042     11      22         0
##   CancellationCode Diverted CarrierDelay WeatherDelay NASDelay SecurityDelay
## 1                        0          339            0        0             0
```

```
## 2                         0          NA          NA          NA          NA
## 3                         0          NA          NA          NA          NA
## 4                         0          NA          NA          NA          NA
## 5                         0          NA          NA          NA          NA
## 6                         0          NA          NA          NA          NA
##   LateAircraftDelay
## 1                 0
## 2                NA
## 3                NA
## 4                NA
## 5                NA
## 6                NA
```

```
## [1] "character"
## [1] "character"
## [1] "character"
## [1] "character"
## [1] "character"
```

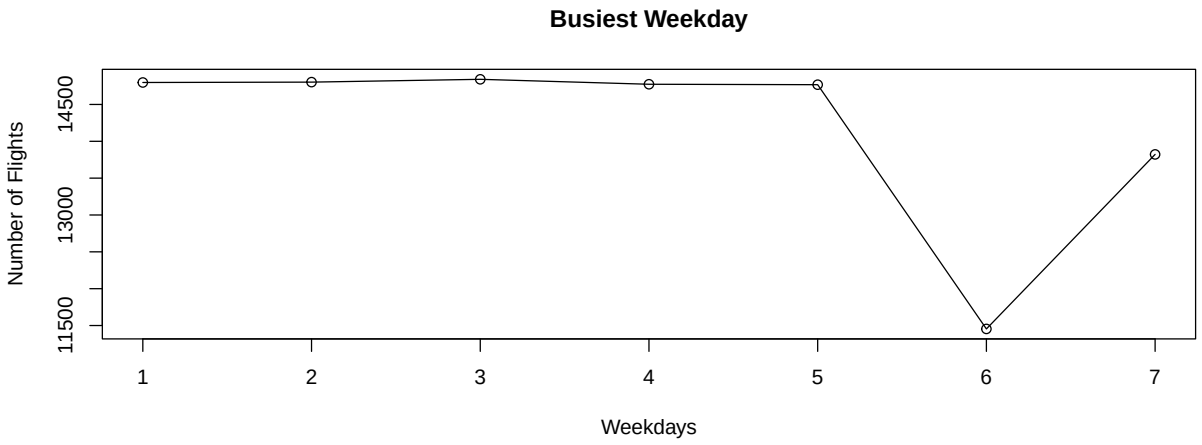
Busiest Month

We will begin by understanding the temporal distribution of flights across the year to find the busiest month. Spring and Summer are the busiest times of the year at the Austin airport with a maximum of about 9,000 flights handled in the month of June.



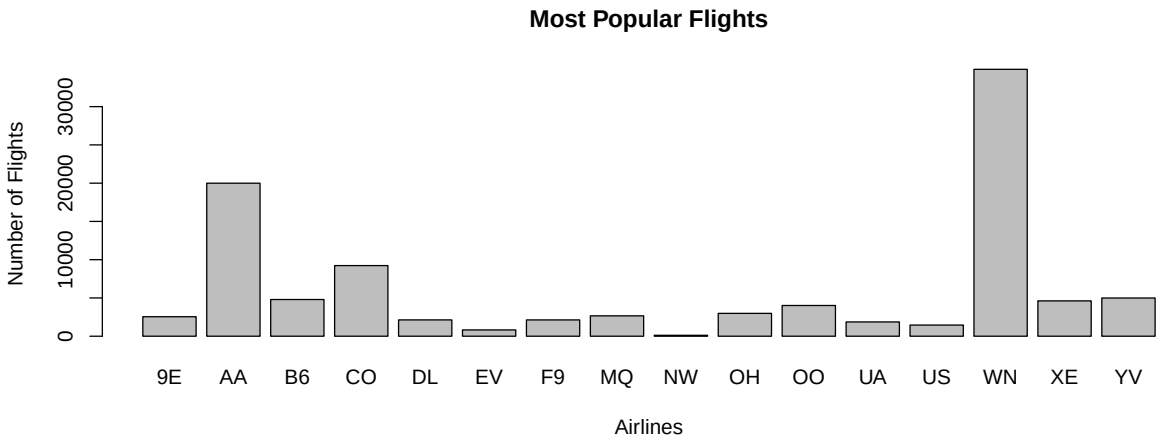
Busiest Weekday

Next up, we will have a look at the temporal distribution of flights across all weeks to find the busiest weekday. All 5 working days, Monday to Friday are equally busy at the airport with a fall in flights on Saturday and a slight bounce back on Sunday.



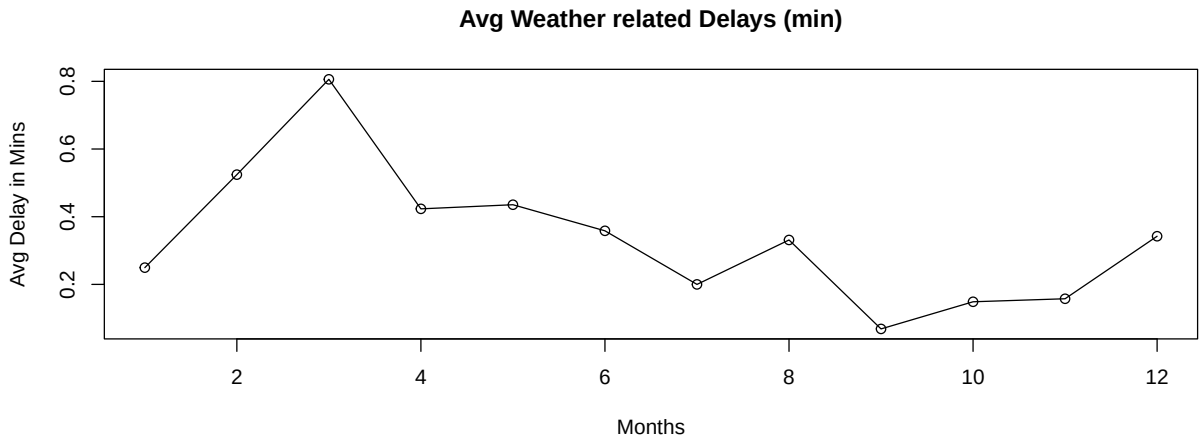
Most Popular Airline

Next up, we look at distribution of flights across various carriers to find the most popular airlines. Southwest Airlines (WN) is the most popular airlines with over 30,000 flights in 2008. This is followed by American Airlines (AA) and Continental Airlines (CO).

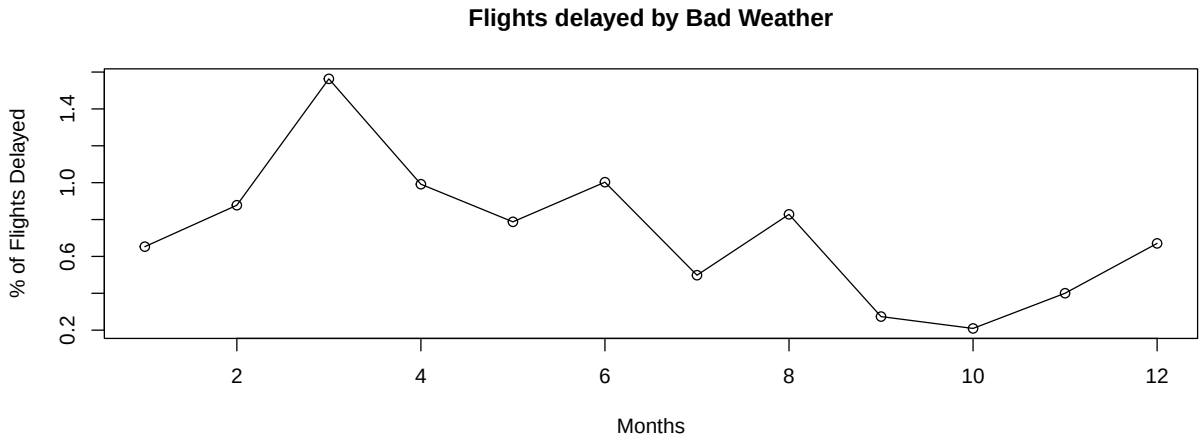


Departure Delays due to Bad Weather

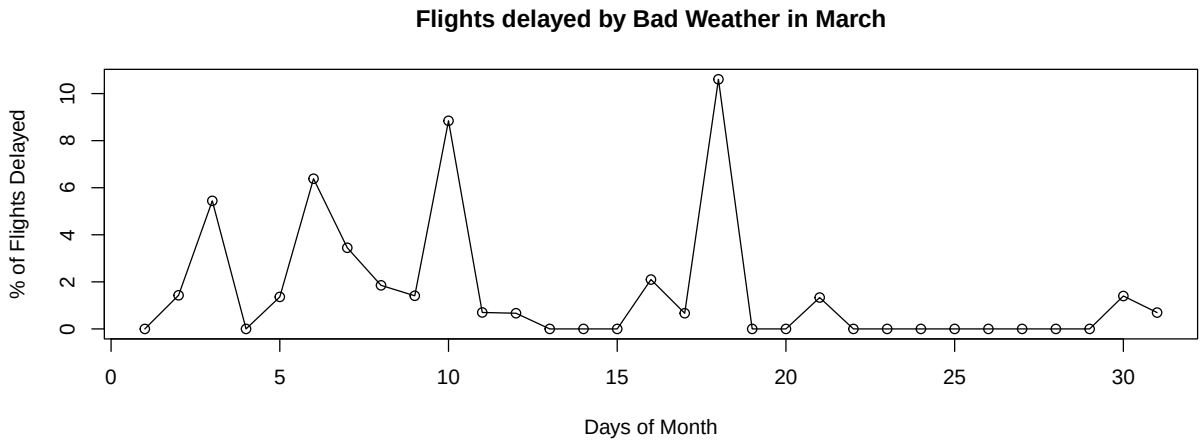
We will analyze the data for delayed departures due to bad weather. We start with checking for delay in departure due to bad weather across multiple months.



While this provides us with average delay in minutes due to bad weather across all months, we want to further see the frequency of weather related delay occurrences in each month.

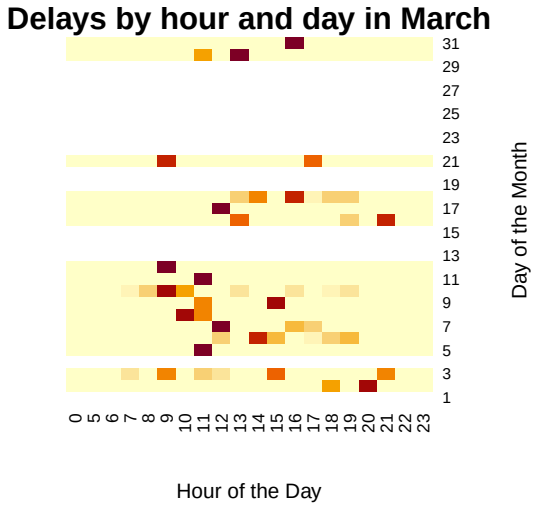


We find similar results to the previous graph with a maximum delays occurring in March. There is 1 weather related delay among every 65 flights in March. With March being the month with most weather related delays, we check for frequency of delays in March.



Since flights were delayed for almost all days across the month (barring a week at the end), the delays cannot be attributed to a standalone weather event. Thus, we can conclude that March has the highest number of weather related delays at the Austin airport. We can further break down weather related delays in March by time of the day to figure out flights at which hours are most likely to be affected by bad weather.

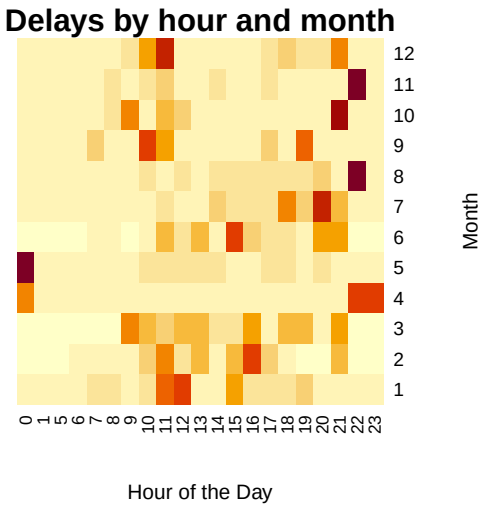
```
## `summarise()` has grouped output by 'DayofMonth'. You can override using the `.groups` argument.
```



We can infer from the heat map that majority of the weather related delays in March have happened before noon and none have happened during the night time. We can plot a similar heat map for all months.

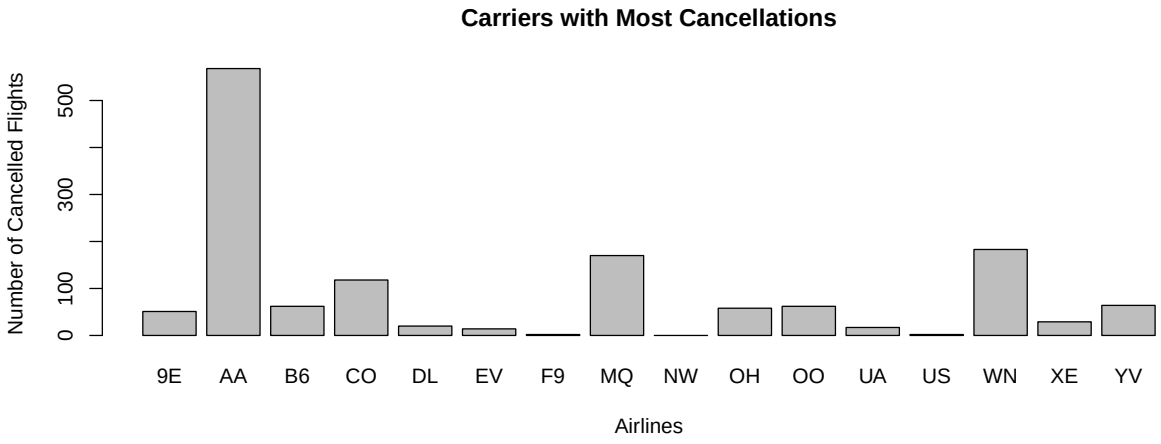
When looking at data for all the months, we still see a trend of no weather related delays during the night hours with delays concentrated in the morning hours right before noon and in the late evening hours.

```
## `summarise()` has grouped output by 'Month'. You can override using the `.groups` argument.
```

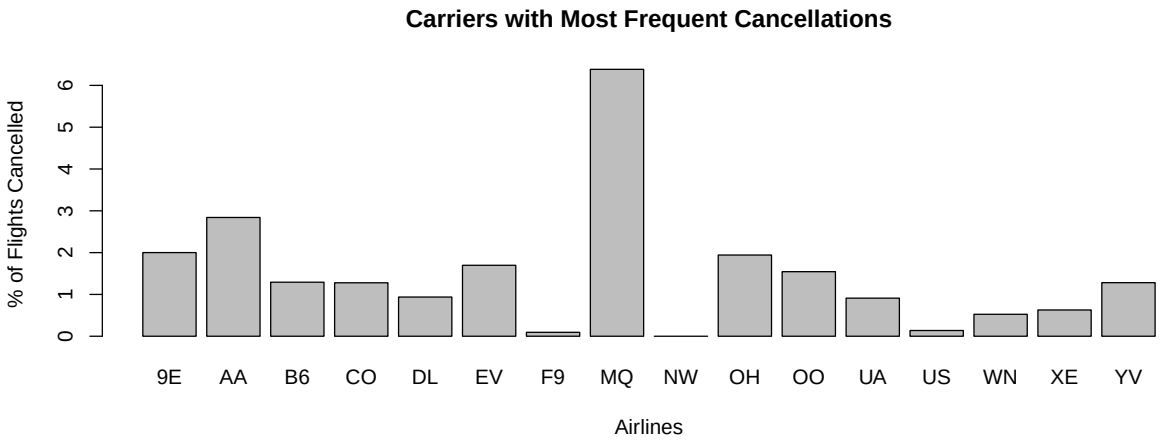


Flight Cancellations

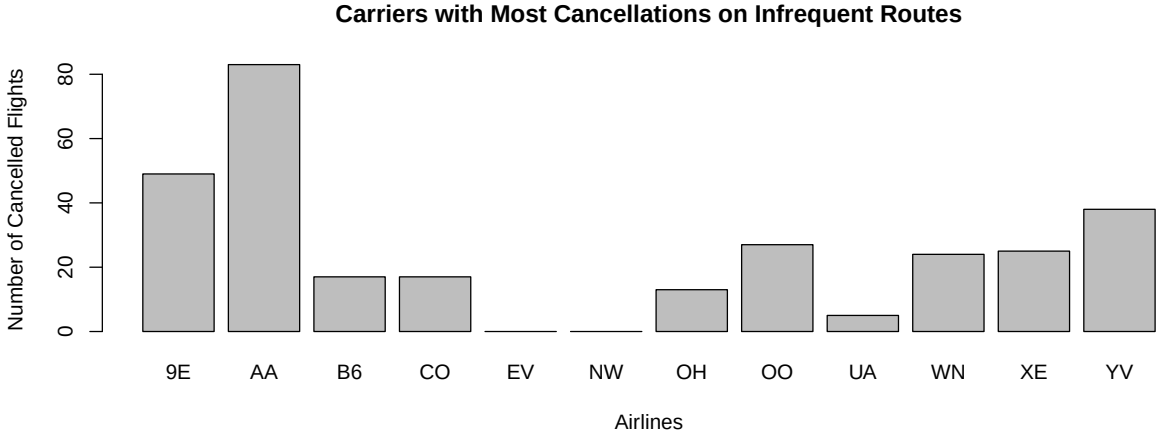
Here, we begin by looking at flight cancellations by various carriers.



American Airlines (AA) has the dubious distinction of being the carrier with most cancelled flights. However, this is an incomplete picture since all carriers may not be flying as many flights in and out of Austin as American. Hence, we look at cancellation frequency among the carriers.

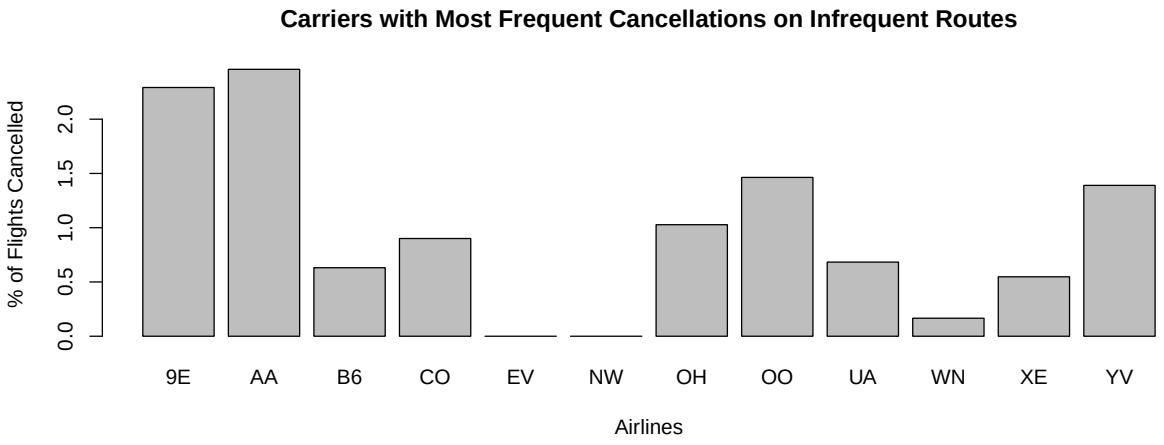


It is now apparent that Envoy Air (MQ) has the highest frequency of cancellations with 1 in every 16 of its flights getting cancelled. Another important metric to look at when studying flight cancellations is frequency of cancellation on routes that do not have more than 1 daily connection. A cancelled flight on such a route means waiting at least a day before the next flight. We will call such routes as *infrequent routes*.



Again, American Airlines (AA) pops up as the carrier with most cancelled flights. We further look at carrier wise flight cancellation frequency for infrequent routes.

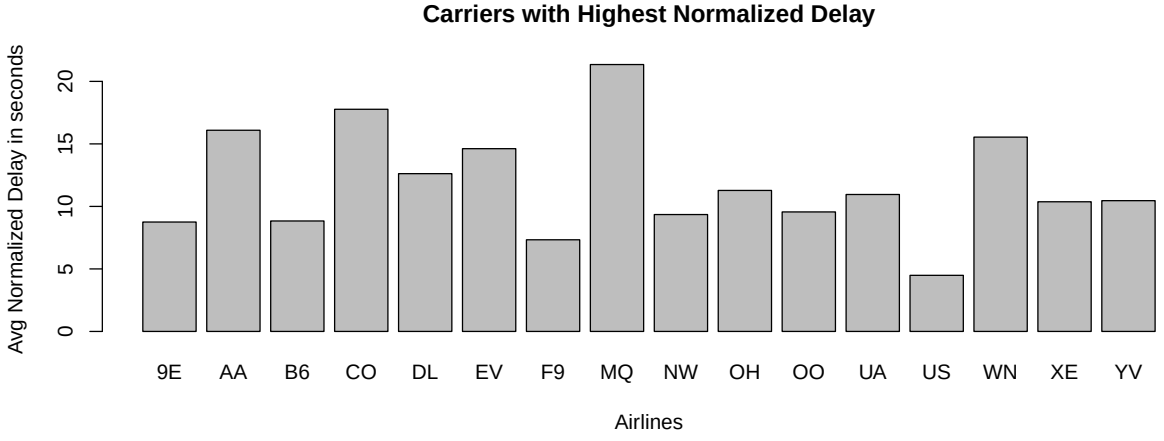
American Airlines (AA) and Endeavor Air (9E) are carriers with highest cancellation frequencies for infrequent routes. Surprisingly, Envoy Air (MQ), which had the worst cancellation rate overall, does not feature here as it does not serve any infrequent route. ExpressJet Airlines (EV) and Northwest Airlines (NW) have had no cancellations serving infrequent routes, thus these carriers can be relied upon when traveling to or from uncommon destinations.



Normalized Delay

Let us now look at Normalized Delay (with respect to total flight time) across multiple variables. Not all delays are equal. A 30-minute delay on a flight to Anchorage is not the same as a 30-minute delay on a short flight like Austin to Dallas or Austin to Houston. We will normalize the delay by dividing it by total flight duration. This gives us delay per unit route duration.

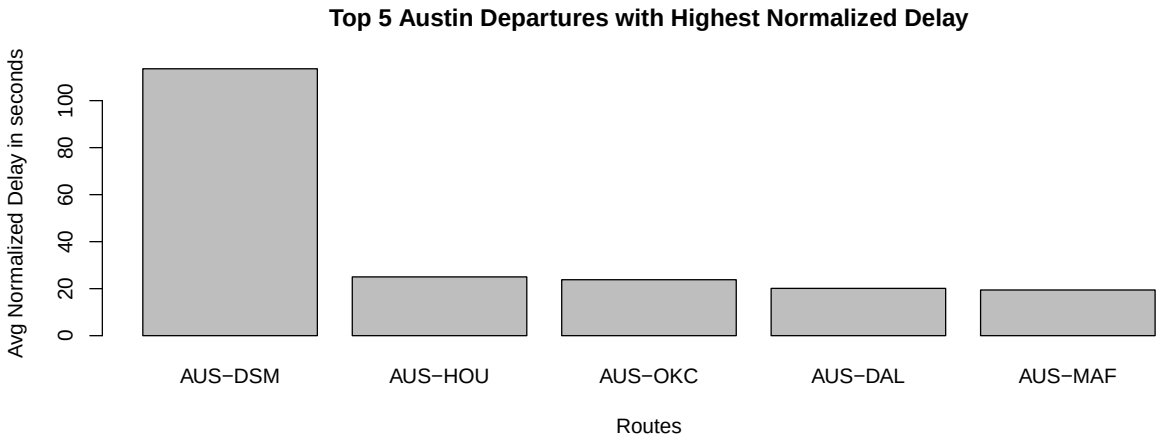
Note: We calculate total normalized delay as sum of normalized departure and arrival delays. While these 2 delays cannot be added directly as a delay in departure will inadvertently lead to delayed arrival, we still proceed with this formula to penalize flights that were not able to compensate their delayed departure in-flight as compared to those flights which had a delayed departure but were able to compensate for the delay by arriving before the estimated time of arrival based on the route flight time.



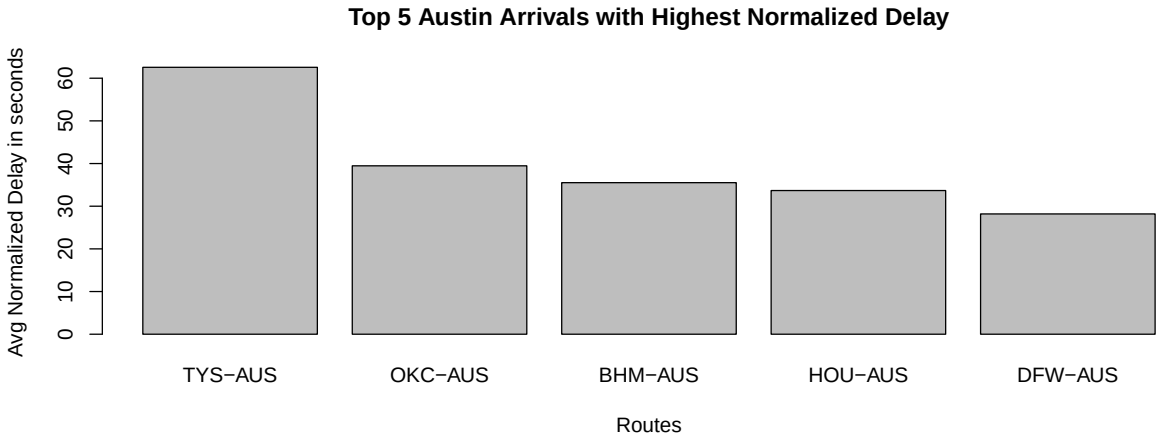
As evident from the bar plot, Envoy Air (MQ) has the highest normalized delays to the tune of 20 seconds for every 1 minute of flight time. US Airways (US) has the least delays. We can further look at routes with the most amount of normalized delays, to and from Austin.

```
## `summarise()` has grouped output by 'Route', 'Origin', 'Dest'. You can override using the `.groups` a
```

1) Routes with highest delays flying out of Austin
The plot throws up Austin to Des Moines as the most delayed route with a delay of 2 minutes for every 1 minute of flight time. This looks fishy, and hence upon further inspection, it turns out that the data has just 1 flight that has flown this route, and the extraordinary delay belongs to this flight. Since we cannot generalize with such a small data sample, Austin to Houston is the next most delayed route with high number of routine flights and an average delay of 20 seconds for every 1 minute of flight time.

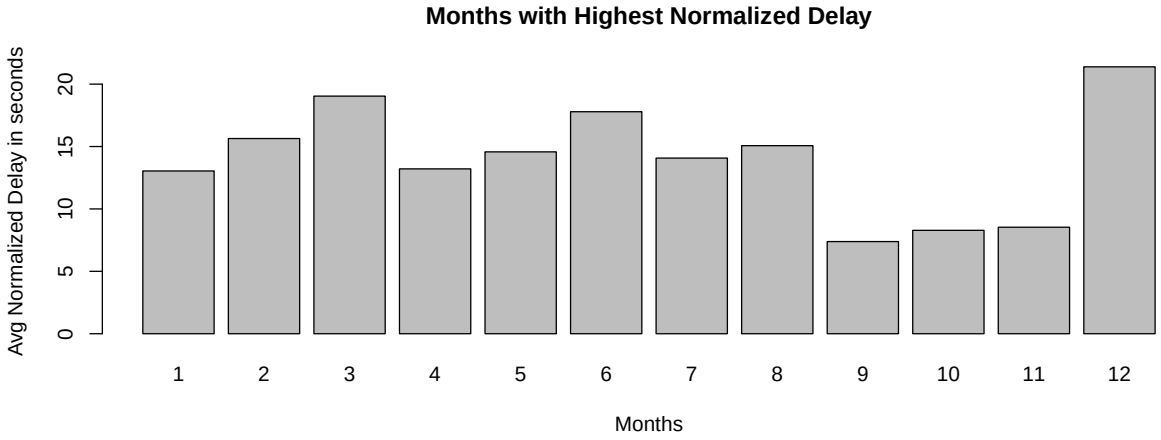


2) Routes with highest delays flying into Austin
As we can see, flights from McGhee Tyson Airport (Knoxville,TN) have the highest delays averaging over 1 minute of delay for every 1 minute of flight time. However, there were only 3 flights from this airport to Austin in the year 2008, and hence the sample is too small to generalize that route as the most delayed at arrival in Austin. Hence, we look at the next route from Oklahoma City to Austin. Flights from Oklahoma City to Austin are delayed by an average of 40 seconds for every 1 minute of flight time. Thus, this is the most delayed route coming in to Austin.
Oklahoma City and Houston have featured in both the plots and thus we can say with confidence that flights to and from Oklahoma City and Houston are the most delayed relative to their flight times.



We can further look at normalized delays across various months.

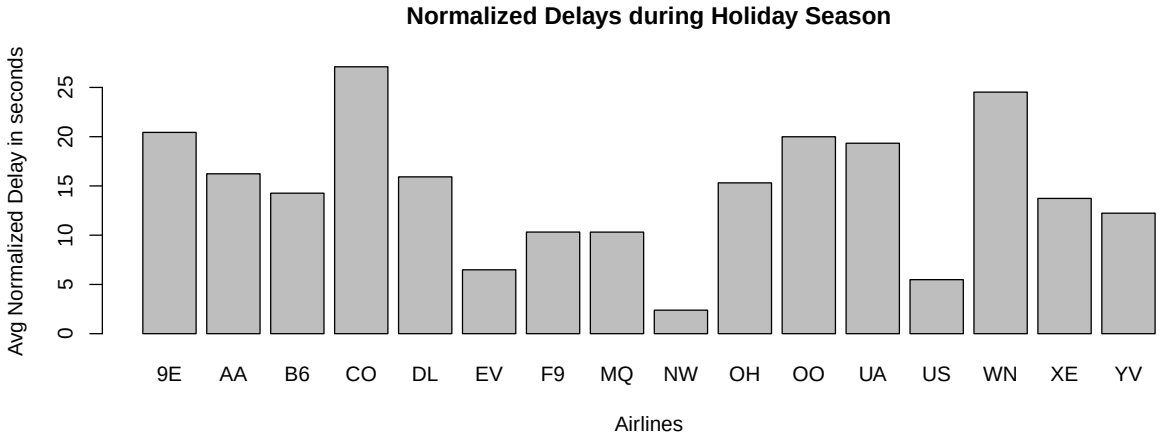
The average normalized delays are the highest in December and March, i.e. during the holiday season and in Spring.



Answering some questions for travelers:

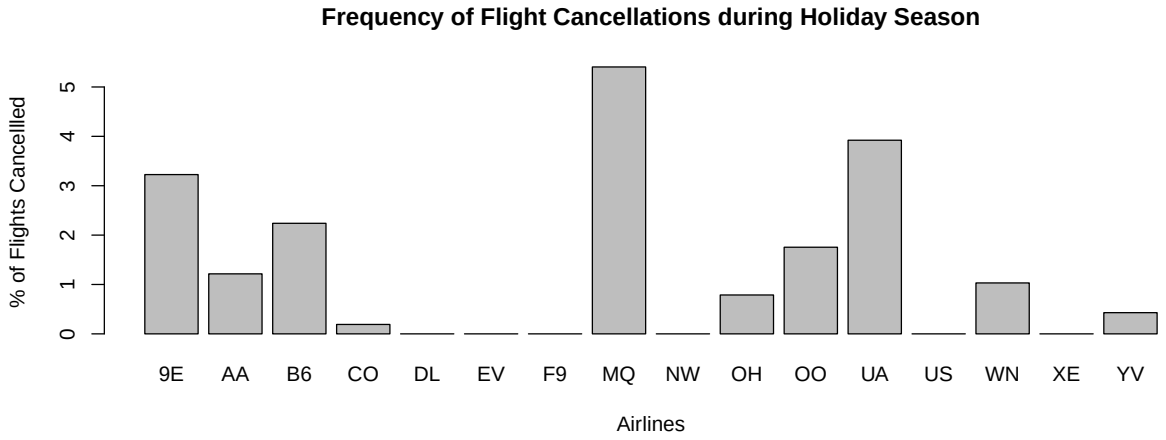
a) Which carriers could be avoided during the Holiday Season?

Any flight that gets cancelled or delayed during the Holiday Season puts a damper on one's spirits. Hence a traveler might want to avoid such flights. We specifically look for cancellations and/or delays between December 20 and January 10 to answer this question.



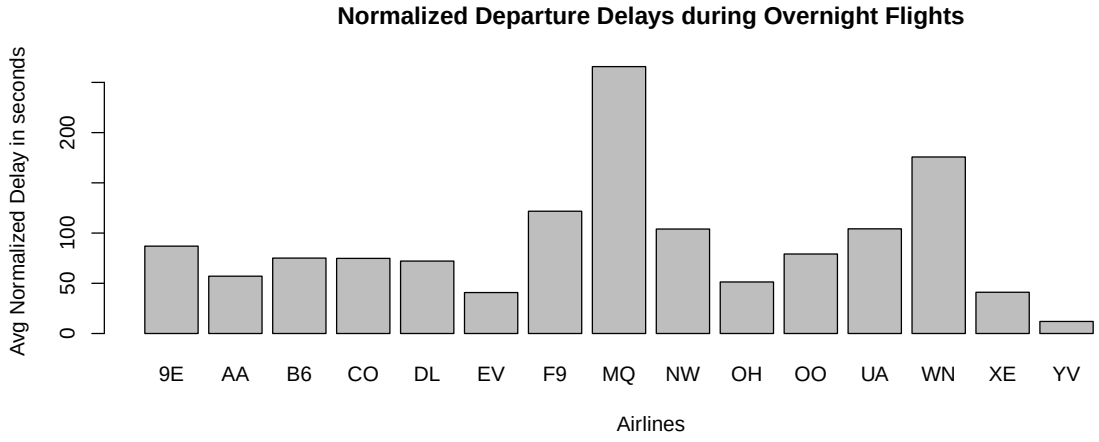
Continental Airlines (CO) has the highest delays, an average of over 25 seconds per 1 minute of flight time. Northwest Airlines (NW) has the least average delay. The observations are different from those we obtained earlier when we had considered the data for all months. Envoy Air (MQ) has been the worst performing carrier overall, but that has been replaced by Continental Airlines (CO) during the holiday season, followed by Southwest Airlines (WN).

Upon looking at flight cancellations, it is evident that Envoy Air (MQ) has the highest cancellations at 5% of their total scheduled flights, followed by United Airlines (UA). Thus some of the carriers to avoid during the holiday season are Continental Airlines, Southwest Airlines, Envoy Air and United Airlines. One can expect a smoother start or end to their holiday travels by hopping onto a flight operated by Northwest Airlines, US Airways or ExpressJet Airlines.



b) Which overnight flights could be avoided?

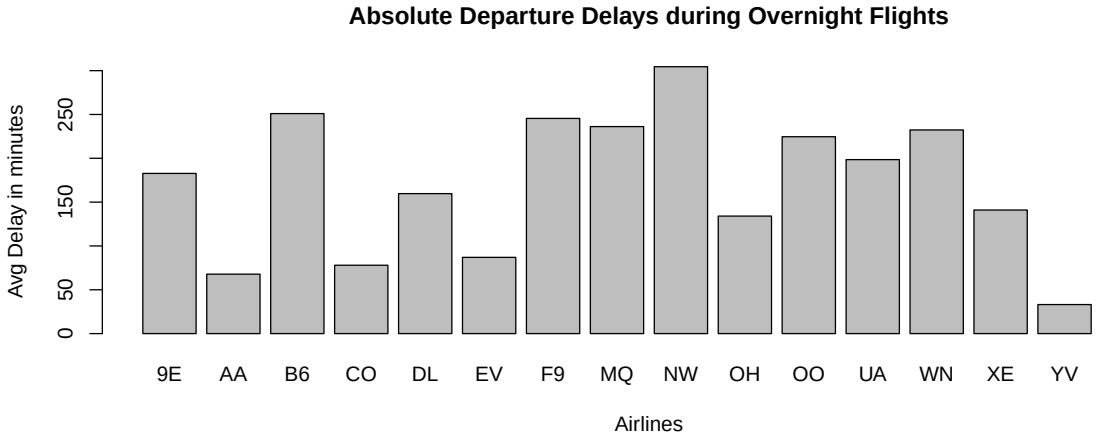
Overnight flights or red-eye flights are the ones that have a substantial part of their journey during the night hours. Travelers generally prefer these flights to avoid spending precious daylight hours in traveling. Any delay in these flights not only throws off the plans, but also causes sleep deprivation as a traveler might be expecting to sleep during these hours in the flight. We look at departure delays in flights that depart between 11:00 PM and 3:59 AM to answer this question.



Envoy Air (MQ) and Southwest Airlines (WN) have the highest normalized delays at departure for overnight flights. Departure delays in overnight flights are a cause of concern irrespective of the duration of the flight that is to be boarded. Thus, we also need to look at absolute departure delays in overnight flights and not just the normalized delays.

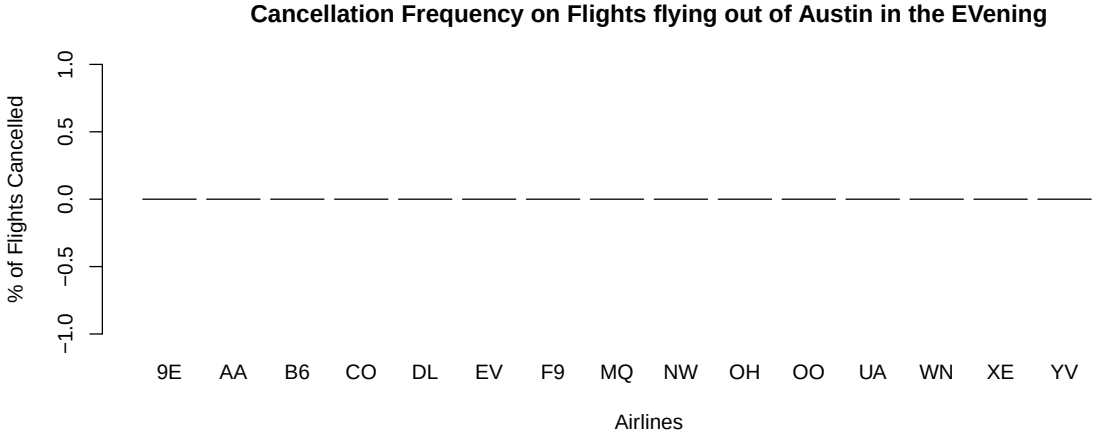
With absolute delays considered, Northwest Airlines (NW) is the worst performing airline with highest average delay per flight at 5 hours, followed by JetBlue Airways (B6) and Frontier Airlines (F9). The best performing airlines for this metric are Mesa Airlines (YV), American Airlines (AA), and Continental Airlines (CO). Envoy Air (MQ) and Southwest Airlines (WN) still show up average delays on the higher side (~4 hours).

Thus, the carriers to avoid on overnight flights are Northwest Airlines, JetBlue Airways, Frontier Airlines, Southwest Airlines and Envoy Air.



c) Which flights could be avoided if you are flying out of Austin at the end of the day?

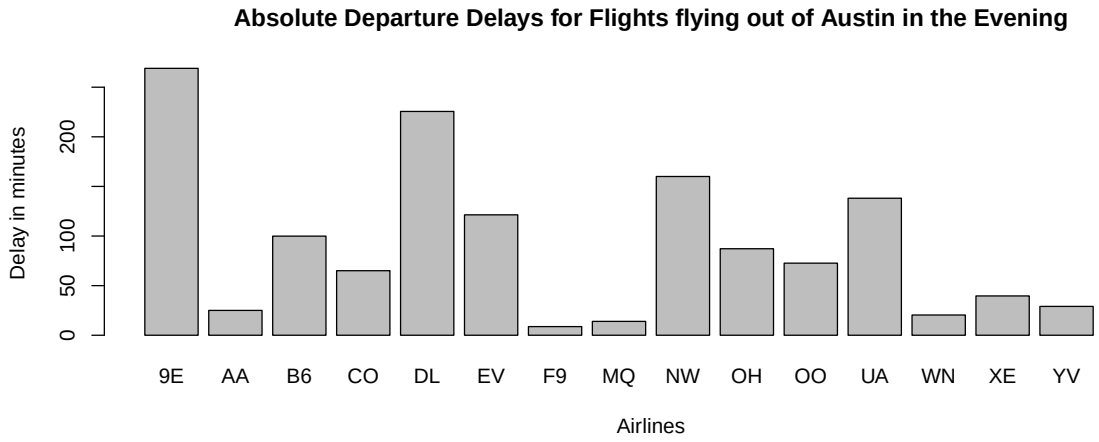
Many travelers take a flight out of a city at the end of the day to save on accommodation costs for the night. A person can spend the entire day in the city and fly out in late evening hours. However, if such a flight is cancelled, the traveler seldom has options apart from staying the night in the city, which entails additional costs for accommodation over and above their planned trip budget. To answer this question, we will be looking at flight cancellations data for departures from Austin between 7:00 PM and 10:59 PM.



It turns out that no flights flying out of Austin during these hours were cancelled in 2008. To further broaden our problem, we will look at delays on flights flying out of Austin during these hours. Again, we

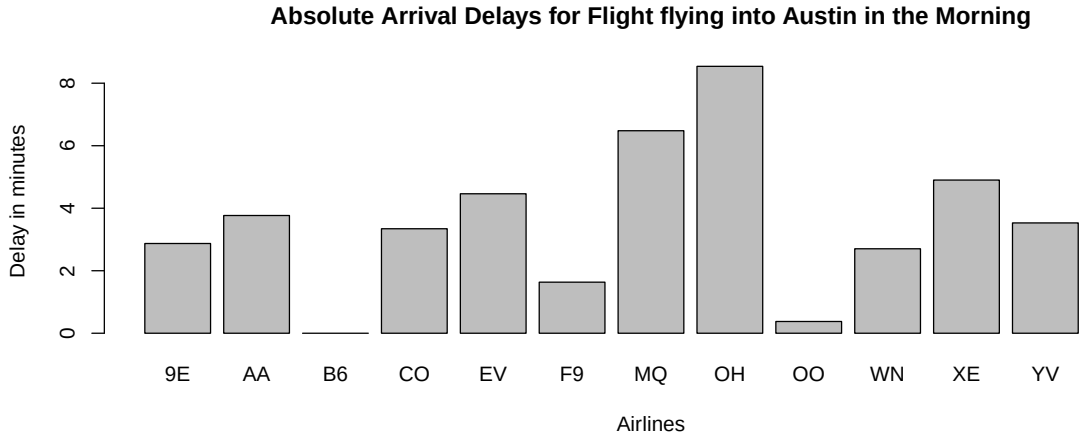
will look at absolute departure delay and not the normalized departure delay as any delay, irrespective of the flight-time upon boarding, is a cause of concern when flying out of Austin at the end of the day.

Endeavor Air (9E) has the highest average delay of about 4 and a half hours per flight when flying out of Austin in late evening. This is followed by Delta Air Lines (DL). These carriers could be avoided when flying out of Austin at the end of the day.



d) Which flights to avoid if flying into Austin for work in the morning?

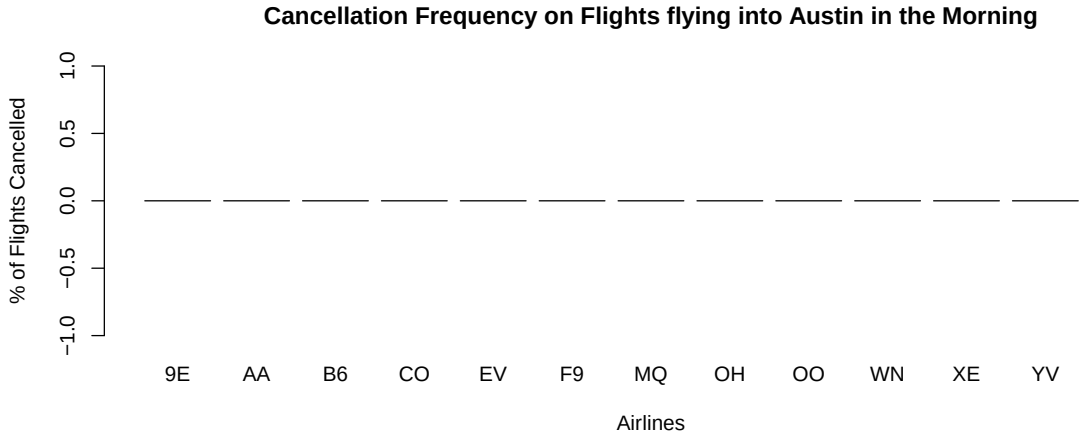
Many professionals that work in Austin fly into the city during the morning hours. Any cancellations or delays on these flights would be detrimental to the objectives of these flyers. Thus, we look at the data for cancellations and delays on flights arriving into Austin between 8:00 AM and 10:59 AM.



PSA Airlines (OH) has the highest average delay of 8 minutes on flights arriving into Austin in the Morning. This is followed by Envoy Air (MQ).

No flights flying into Austin in the Morning hours were cancelled in 2008. Thus, based on delays

alone, professionals can avoid flying PSA Airlines and Envoy Air when flying into Austin in the Morning hours. However, the delays here are not as significant as those seen during other hours of the day.



Problem 3: Portfolio Modeling

In this problem, we will construct three different portfolios of exchange-traded funds, or ETFs, and use bootstrap resampling to analyze the short-term tail risk of our portfolios.

```
## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

##
## Attaching package: 'xts'

## The following objects are masked from 'package:dplyr':
##
##     first, last

## Loading required package: TTR

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

##
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

We selected the ETFs ensuring diversity and different levels of risk.

Invesco QQQ is an exchange-traded fund that tracks the Nasdaq-100 IndexT. The Index includes the 100 largest non-financial companies listed on the Nasdaq. It is one of the largest, owns only non-financial stocks and is tech-heavy. The stock had performed well in 2017 but had a poor return in 2018.

The SPDR S&P 500 trust is an exchange-traded fund which trades on the NYSE Arca under the symbol. SPDR is an acronym for the Standard & Poor's Depositary Receipts, the former name of the ETF. It is designed to track the S&P 500 stock market index. SPY is one of the safest and largest ETFs around.

SVXY is the ProShares Short VIX Short-Term Futures ETF, which provides investors exposure to short VIX futures contracts. Put simply, investors who buy SVXY are short S&P 500 volatility futures. It is a high risk ETF. This is an unusual ETF since the performance is dependent on the market volatility, not security.

ProShares UltraShort FTSE Europe (EPV) is a low performing ETF for the past few years. iShares Core Growth Allocation ETF (AOR) is a very diverse ETF. The iShares Core Growth Allocation ETF seeks to track the investment results of an index composed of a portfolio of underlying equity and fixed income funds intended to represent a growth allocation target risk strategy.

YYY is a portfolio of 45 closed-end funds (CEFs) based on a rules based index. This investment approach results in a portfolio which contains a variety of asset classes, investment strategies and asset managers. The index seeks to measure the performance of the top 45 U.S. exchange-listed closed-end funds.

In total, we have selected 6 ETFs - "QQQ", "SPY", "SVXY", "EPV", "AOR" and "YYY". We have considered 5 years of ETF data starting from 01-Jan-2014.

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.

## pausing 1 second between requests for more than 5 symbols
## pausing 1 second between requests for more than 5 symbols

## [1] "QQQ"  "SPY"  "SVXY" "EPV"  "AOR"  "YYY"

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/QQQ?
## period1=-2208988800&period2=1629072000&interval=1d&events=split&crumb=dsaLzY2dWDR'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/QQQ?
## period1=-2208988800&period2=1629072000&interval=1d&events=split&crumb=dsaLzY2dWDR'
```

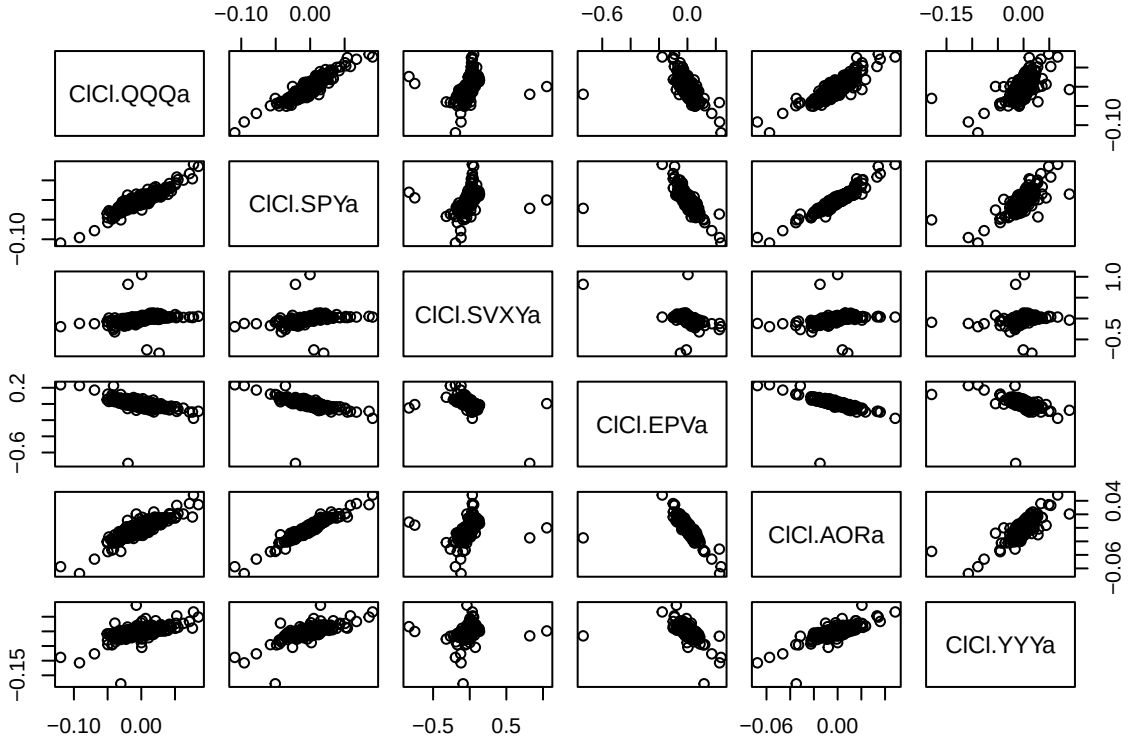
```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/SPY?
## period1=-2208988800&period2=1629072000&interval=1d&events=split&crumb=dsaLzY2dWDR'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/SPY?
## period1=-2208988800&period2=1629072000&interval=1d&events=split&crumb=dsaLzY2dWDR'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/SVXY?
## period1=-2208988800&period2=1629072000&interval=1d&events=div&crumb=dsaLzY2dWDR'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/SVXY?
## period1=-2208988800&period2=1629072000&interval=1d&events=split&crumb=dsaLzY2dWDR'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/SVXY?
## period1=-2208988800&period2=1629072000&interval=1d&events=split&crumb=dsaLzY2dWDR'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/EPV?
## period1=-2208988800&period2=1629072000&interval=1d&events=split&crumb=dsaLzY2dWDR'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/EPV?
## period1=-2208988800&period2=1629072000&interval=1d&events=split&crumb=dsaLzY2dWDR'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/AOR?
## period1=-2208988800&period2=1629072000&interval=1d&events=split&crumb=dsaLzY2dWDR'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/AOR?
## period1=-2208988800&period2=1629072000&interval=1d&events=split&crumb=dsaLzY2dWDR'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/YYY?
## period1=-2208988800&period2=1629072000&interval=1d&events=split&crumb=dsaLzY2dWDR'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/YYY?
## period1=-2208988800&period2=1629072000&interval=1d&events=split&crumb=dsaLzY2dWDR'
```

Sample Data for YYY

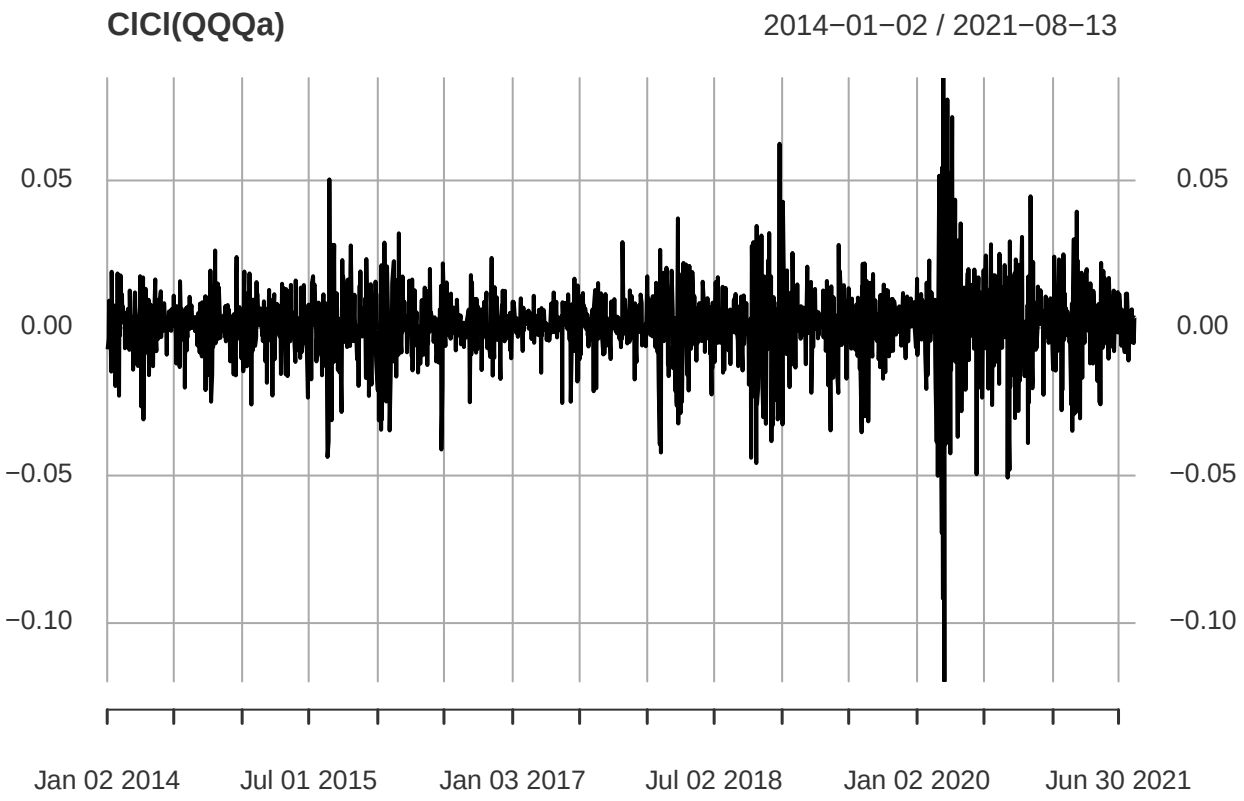
```
##            YYY.Open YYY.High  YYY.Low YYY.Close YYY.Volume YYY.Adjusted
## 2014-01-02 11.56647 11.56647 11.50165  11.51163      19600     11.51163
## 2014-01-03 11.58641 11.58641 11.44681  11.53157      15900     11.53157
## 2014-01-06 11.50664 11.54154 11.49667  11.54154       9800     11.54154
## 2014-01-07 11.54154 11.62629 11.54154  11.62629      10700     11.62630
## 2014-01-08 11.61632 11.61632 11.56148  11.57644      10400     11.57644
## 2014-01-09 11.54652 11.58641 11.54154  11.58641      13500     11.58641
```

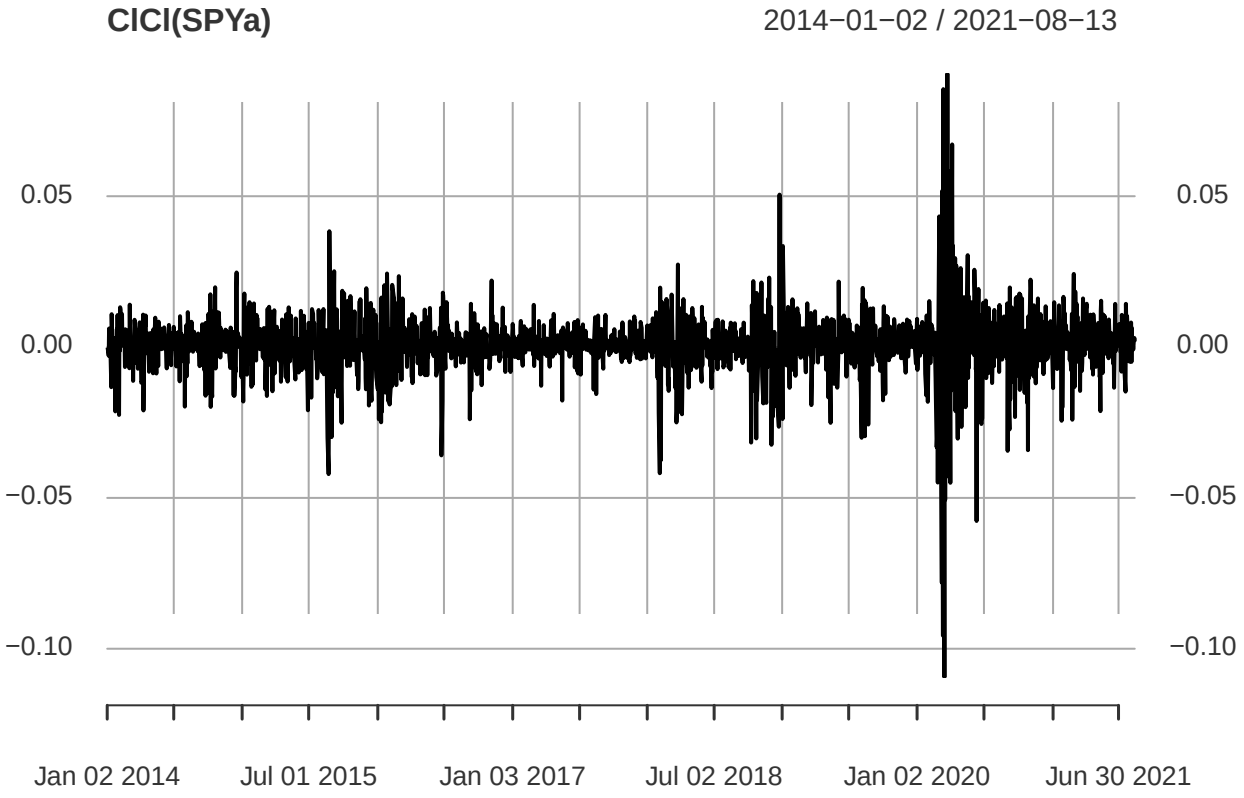
```
##               ClCl.QQQa     ClCl.SPYa   ClCl.SVXYa     ClCl.EPVa     ClCl.AORa
## 2014-01-03 -0.007218953 -0.0001640007  0.006050087 -0.0006365532  0.0010468726
## 2014-01-06 -0.003693433 -0.0028979059  0.011952161 -0.0006369586 -0.0020915556
## 2014-01-07  0.009267875  0.0061416703  0.023399265 -0.0146589871  0.0060257797
## 2014-01-08  0.002180842  0.0002180510 -0.001669529  0.0025873221 -0.0026042447
## 2014-01-09 -0.003321510  0.0006538524 -0.002617421 -0.0012903548 -0.0007832637
## 2014-01-10  0.003217720  0.0027227184  0.023472818 -0.0206718191  0.0065325320
##                ClCl.YYYa
## 2014-01-03  0.0017323084
## 2014-01-06  0.0008647212
## 2014-01-07  0.0073434125
## 2014-01-08 -0.0042882075
## 2014-01-09  0.0008613695
## 2014-01-10  0.0038726334
```

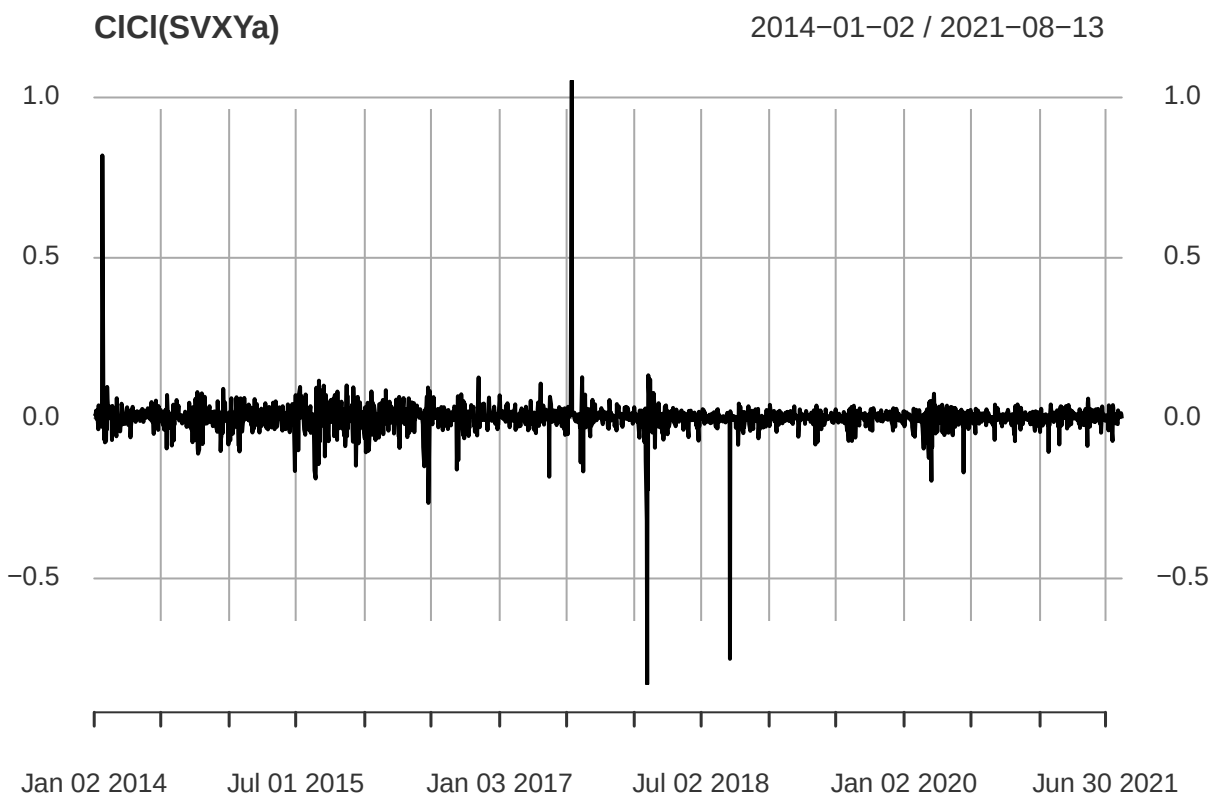
Lets look at how the stocks are performing relative to each other. We can see a strong correlation here. But it is complex and non-linear. As discussed above, a few are performing well, others are not.

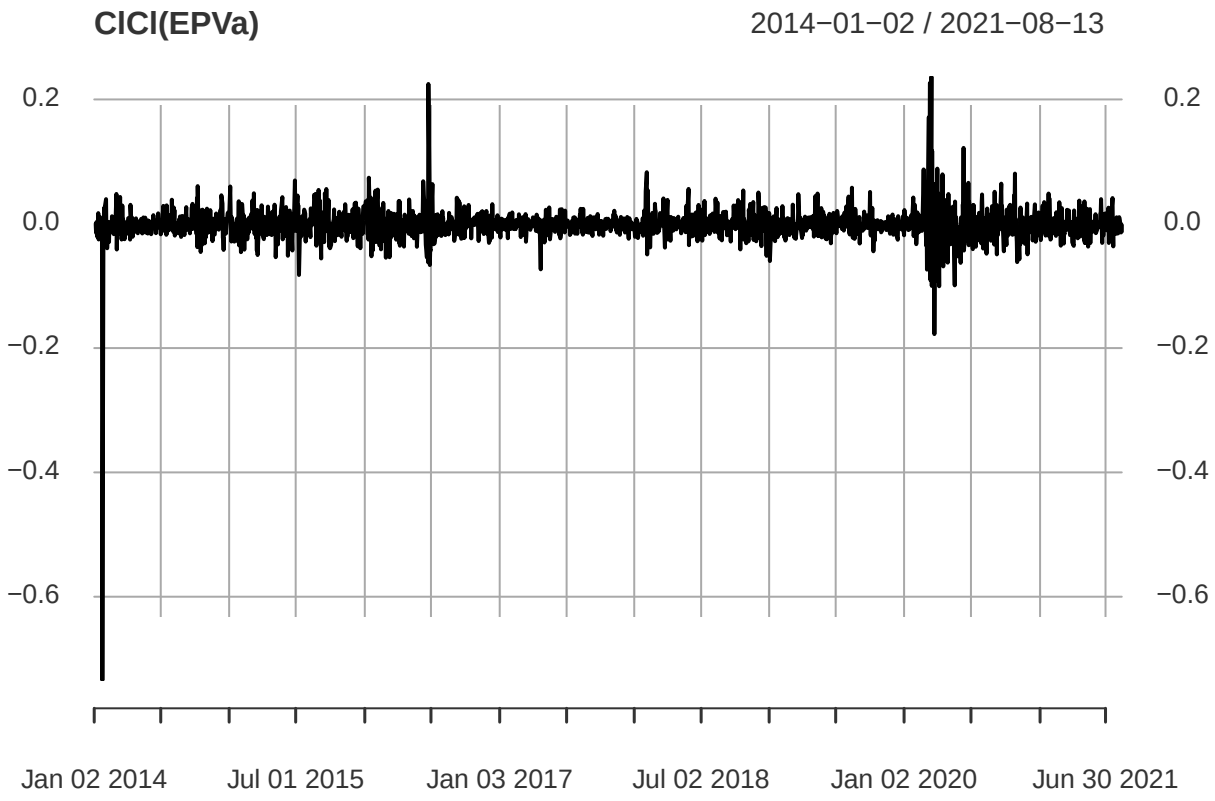


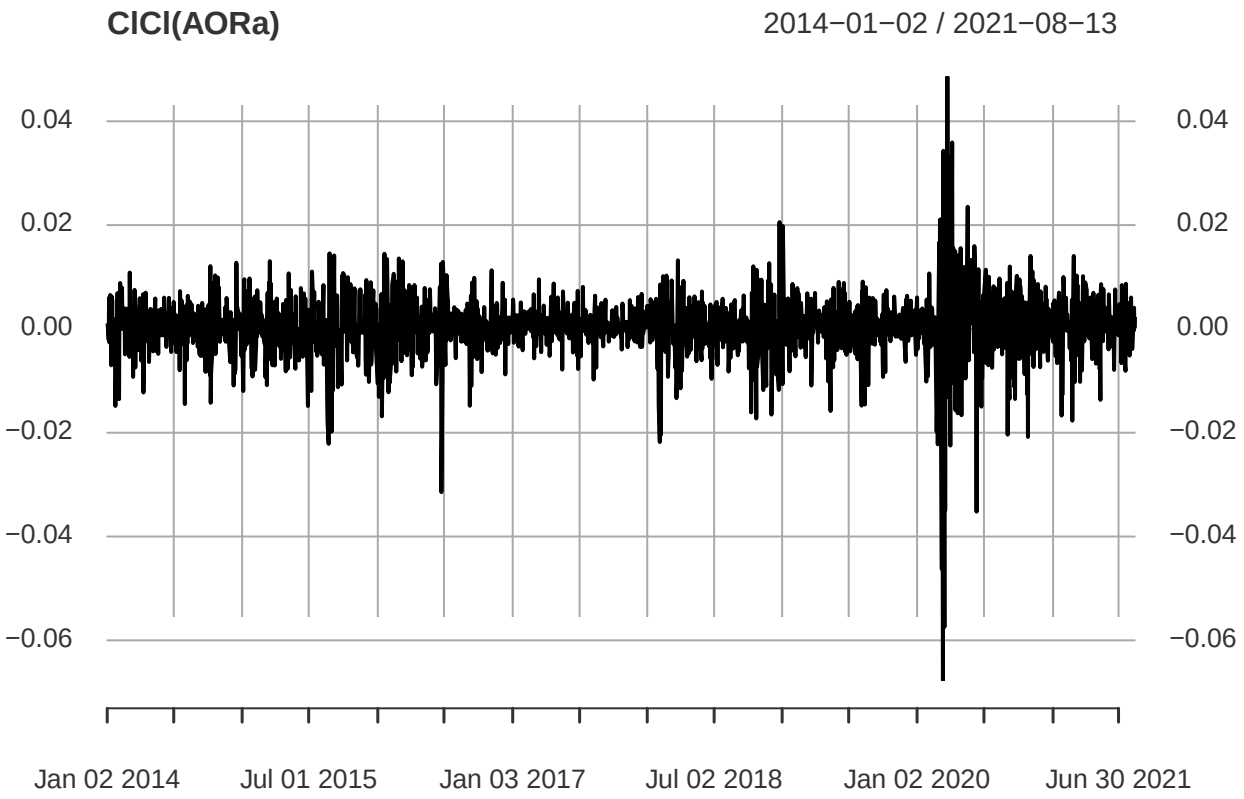
Volatility of the ETFs across the 5 year period.

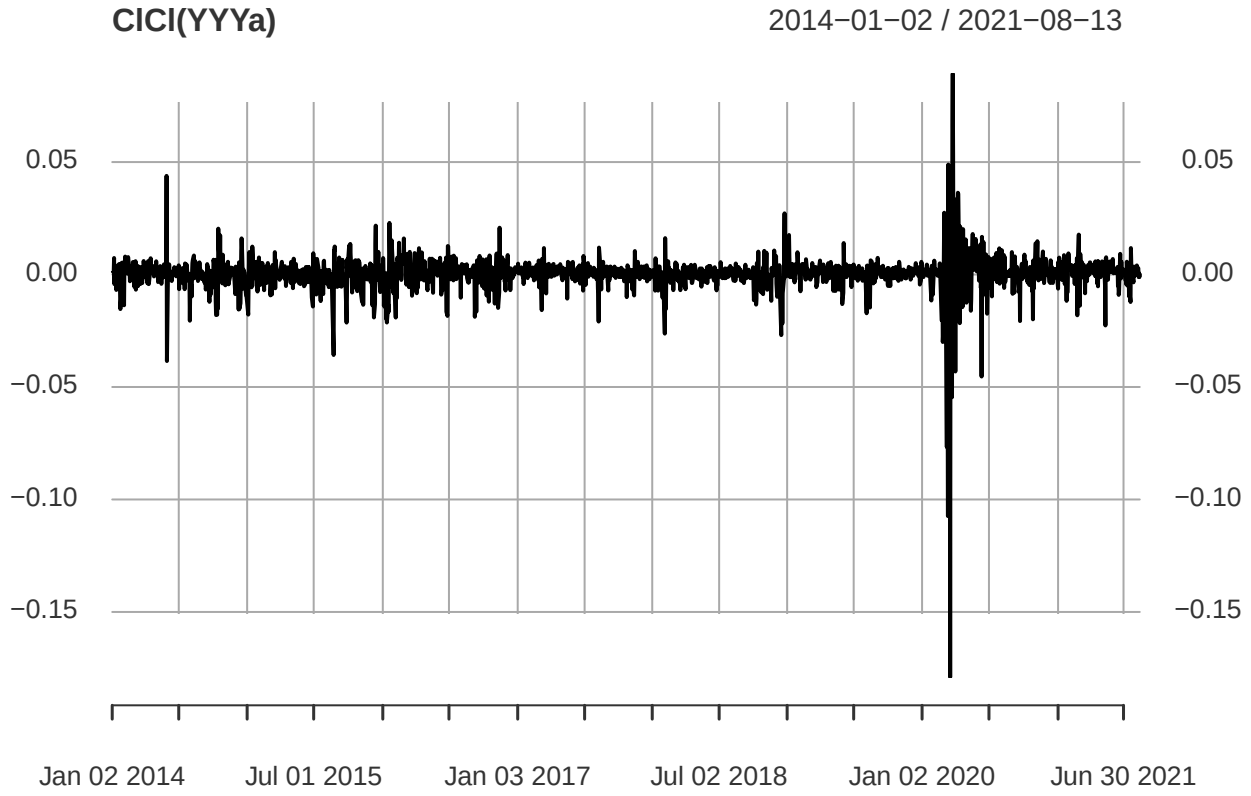



ClCl(SPYa)
2014−01−02 / 2021−08−13
0.05
0.00
−0.05
−0.10
Jan 02 2014
Jul 01 2015
Jan 03 2017
Jul 02 2018
Jan 02 2020
Jun 30 2021


ClCl(SVXYa)
2014−01−02 / 2021−08−13
1.0
0.5
0.0
−0.5
Jan 02 2014
Jul 01 2015
Jan 03 2017
Jul 02 2018
Jan 02 2020
Jun 30 2021


ClCl(EPVa)
2014−01−02 / 2021−08−13
0.2
0.0
−0.2
−0.4
−0.6
Jan 02 2014
Jul 01 2015
Jan 03 2017
Jul 02 2018
Jan 02 2020
Jun 30 2021


ClCl(AORa)
2014−01−02 / 2021−08−13
0.04
0.02
0.00
−0.02
−0.04
−0.06
Jan 02 2014
Jul 01 2015
Jan 03 2017
Jul 02 2018
Jan 02 2020
Jun 30 2021



We can clearly observe while some stocks are highly volatile while some are more stable in nature hence resulting in a balanced overall selection

Our initial wealth is $100,000

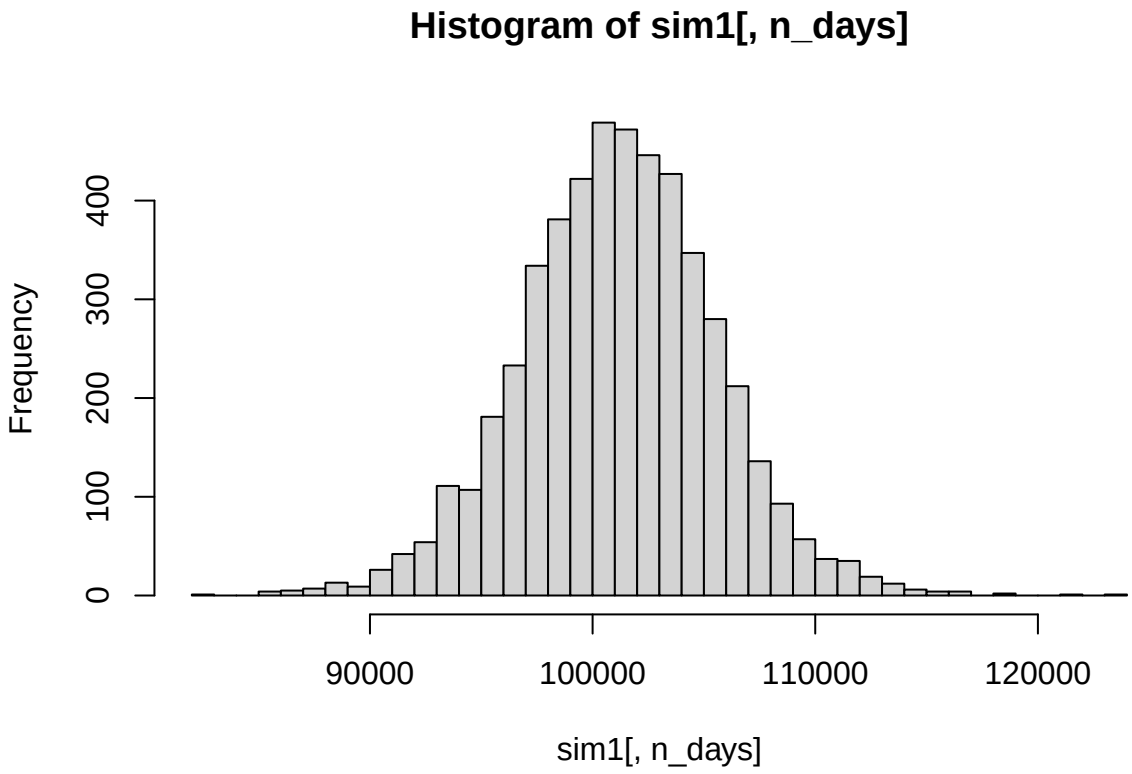
SIMULATION 1 : SAFE Portfolio

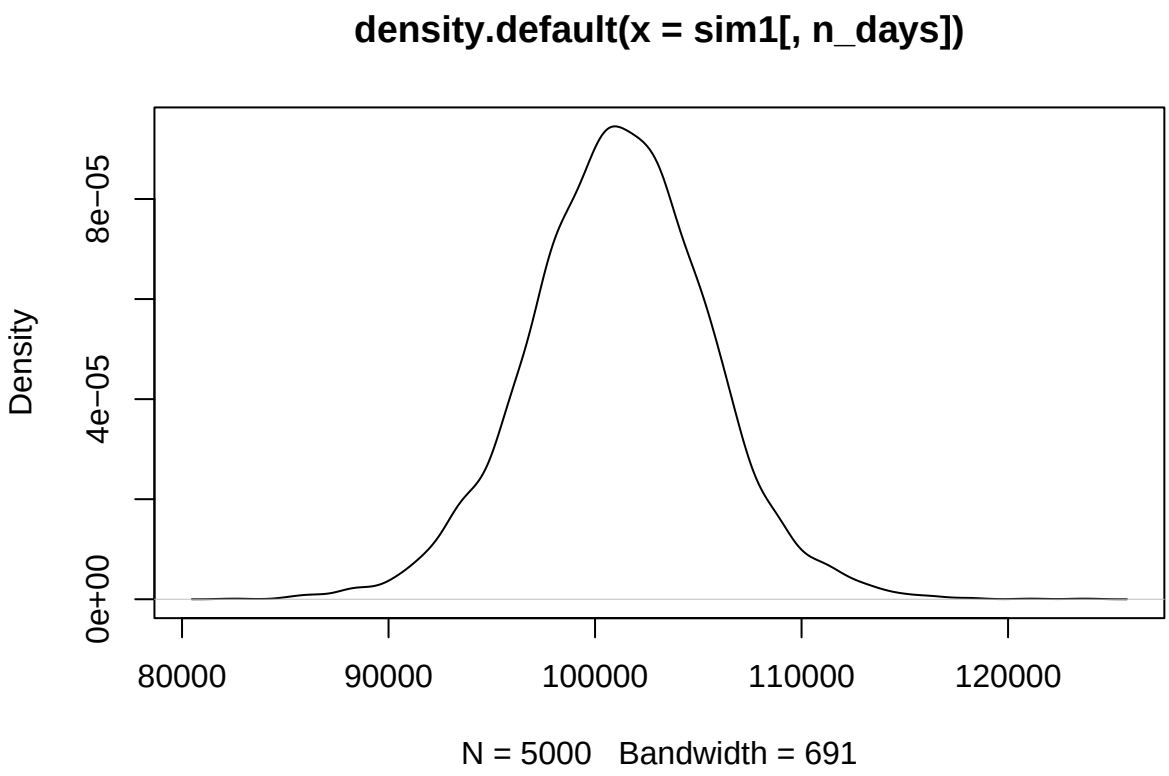
ETFs: “QQQ”, “SPY”, “SVXY”, “EPV”, “AOR”, “YYY”

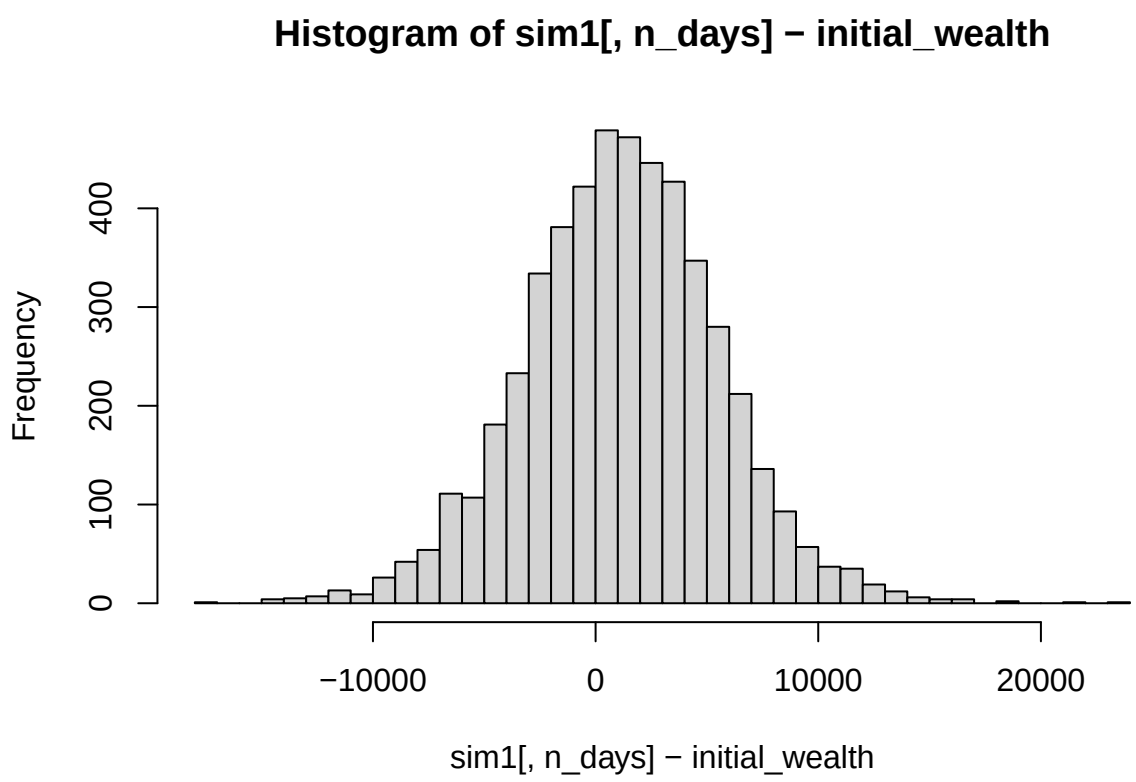
For the safe portfolio, we distributed 90% of the total wealth among the high performing ETFs - QQQ, SPY and AOR.

```
##              [,1]     [,2]      [,3]     [,4]      [,5]      [,6]      [,7]
## result.1 100816.1 100802.3 100858.93 100982.5 100429.94 100638.91 100838.42
## result.2 100385.8 100298.9  98105.83  98129.7  96562.64  97141.92  97112.21
## result.3 100424.0 100601.7  99550.25 100254.8  98062.69  98618.82  96935.84
## result.4 100647.2 102054.3 102516.97 102993.7 102996.70 102671.17 102748.14
## result.5 100411.6 100291.6 100791.19 101500.8 101690.36 101620.27 102003.30
## result.6 100124.7 100711.6 100205.81 100067.2 100226.22 101888.07 102964.97
##               [,8]      [,9]     [,10]     [,11]     [,12]     [,13]     [,14]
## result.1 100701.42 103909.31 103744.55 103928.13 103834.54 103468.33 105152.55
## result.2  96900.19  97287.41  97073.78  97680.31  96769.17  98237.29  99208.78
## result.3  97083.24  96144.79  96404.00  96182.21  96504.97  96357.97  95887.81
## result.4 102723.73 102597.53 104792.85 102309.08 102013.95  99899.73 106624.31
## result.5 100756.07 100551.24 101092.19 101855.41 100799.32 100512.06 100363.75
## result.6 102682.15 101348.63 102861.16 103141.25 104194.94 103190.40 103186.35
##              [,15]     [,16]     [,17]     [,18]    [,19]     [,20]
## result.1 105298.82 105636.41 105390.55 105617.52 106350.8 106342.87
```

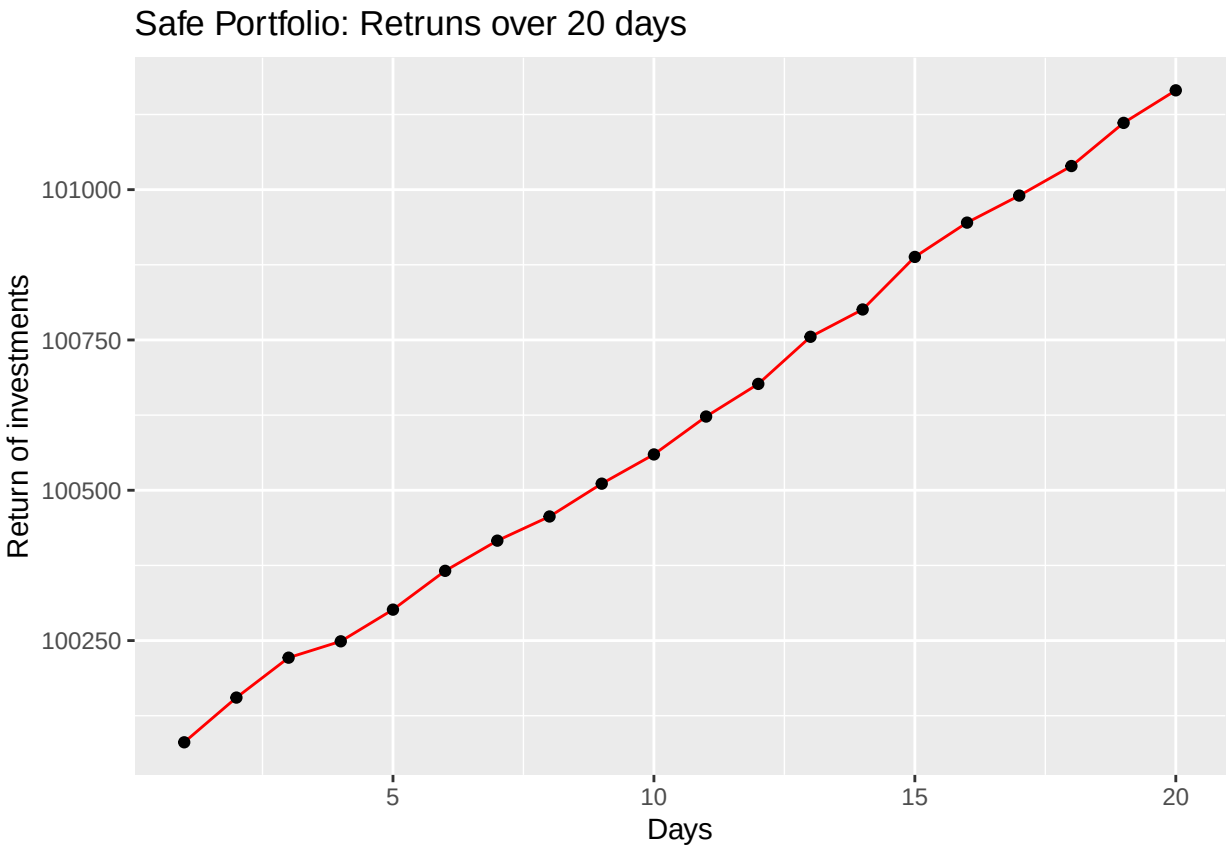
```
## result.2  99795.92 100253.24 100935.97 100333.01 100266.1  99770.75
## result.3  96701.30  97576.82  98562.01  97738.94  97788.4  97830.29
## result.4 106767.56 106302.97 106855.35 106724.08 106702.7 106574.20
## result.5 100568.67 102037.22 102633.11 102892.59 100107.2 100093.66
## result.6 103508.26 103198.34 100095.94 100262.42 100791.2 101509.47
```

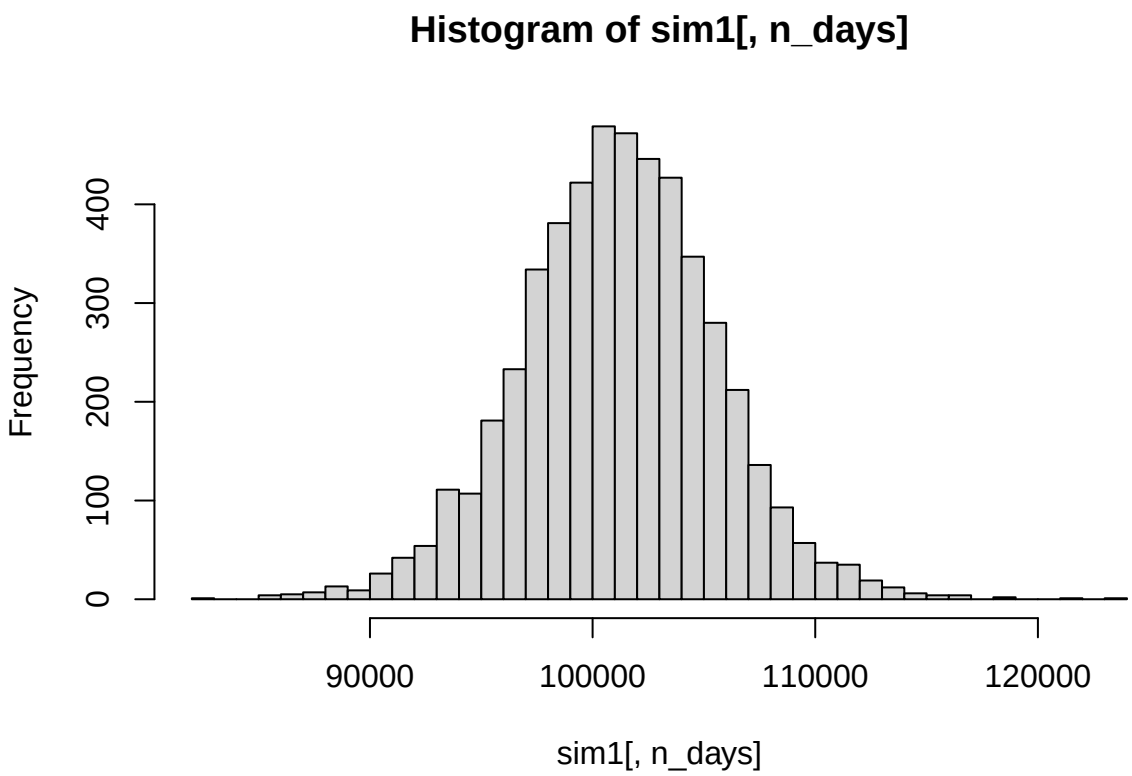


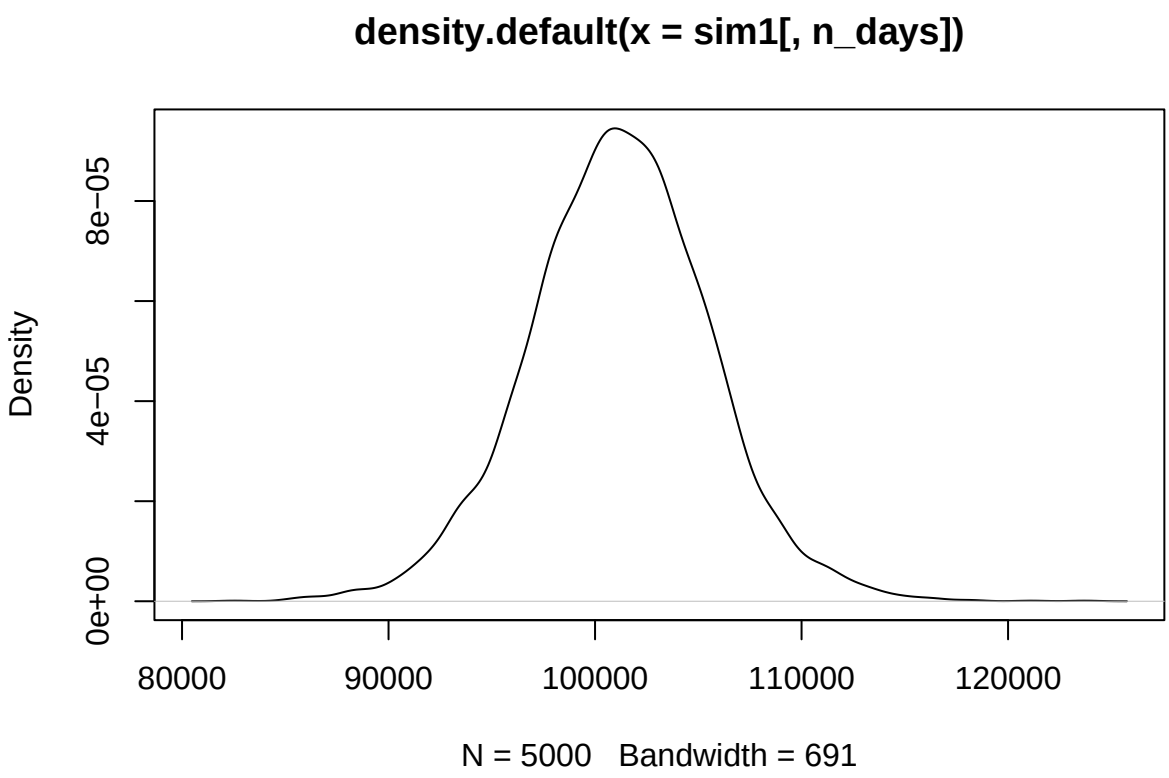

density.default(x = sim1[, n_days])
Density
0e+00
4e−05
8e−05
80000
90000
100000
110000
120000
N = 5000 Bandwidth = 691

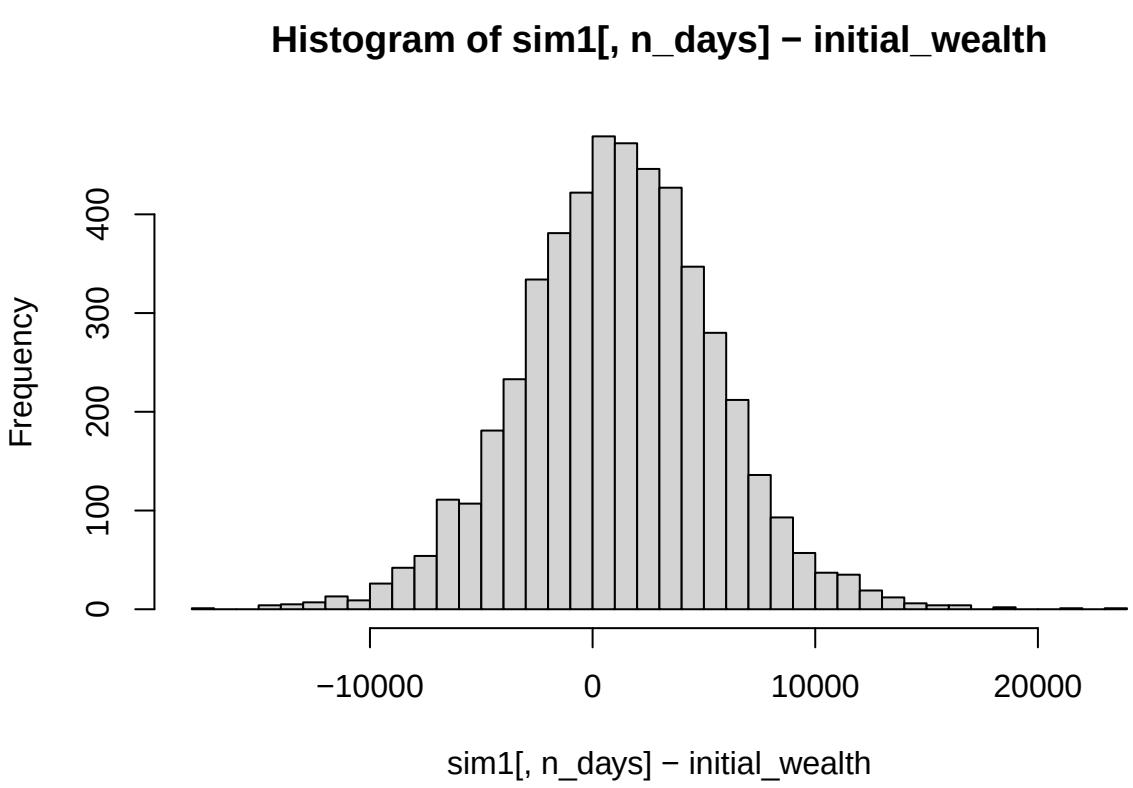



```
## Confidence Interval from Bootstrap Distribution (5000 replicates)

##
## Average return of investement after 20 days 101165.2

##
## 5% Value at Risk for safe portfolio- -6173.529
```


Safe Portfolio: Retruns over 20 days
Return of investments
101000
100750
100500
100250
5
10
15
20
Days


Histogram of sim1[, n_days]
Frequency
0
100
200
300
400
90000
100000
110000
120000
sim1[, n_days]


density.default(x = sim1[, n_days])
Density
0e+00
4e−05
8e−05
80000
90000
100000
110000
120000
N = 5000 Bandwidth = 691




```
## Confidence Interval from Bootstrap Distribution (5000 replicates)

##
## Average return of investement after 20 days 101165.2

##
## 5% Value at Risk for safe portfolio- -6173.529
```

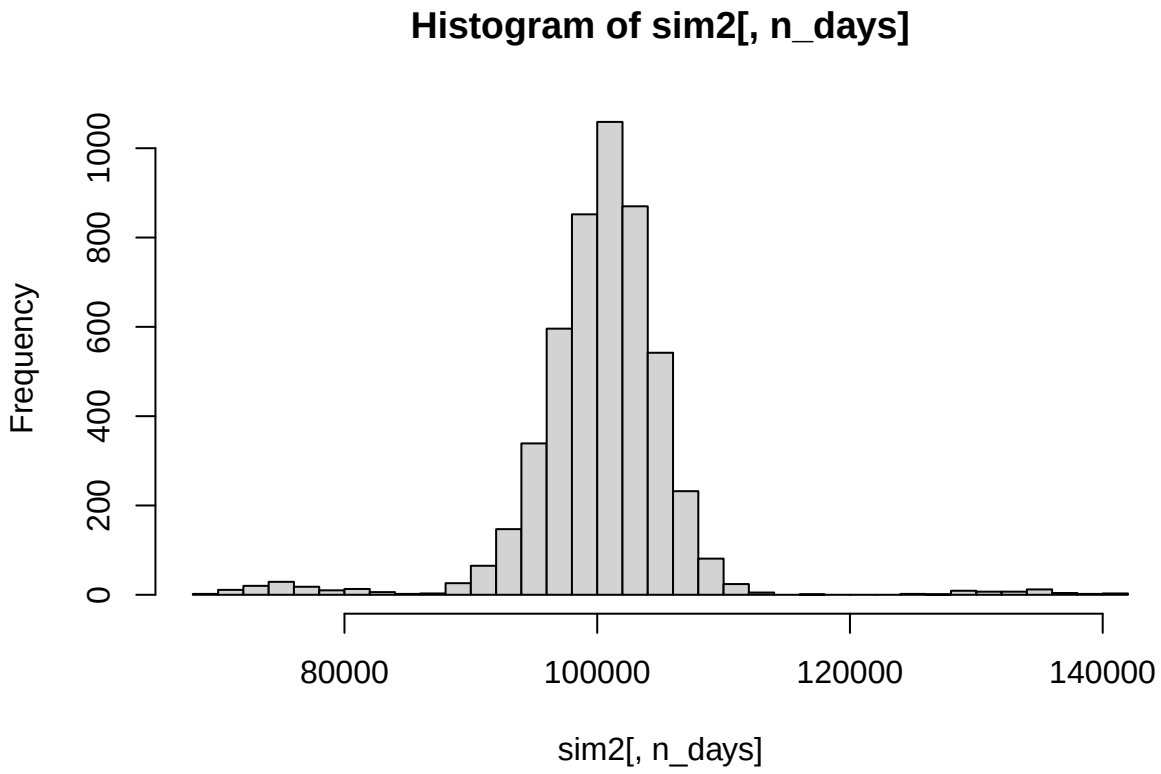
SIMULATION 2 : HIGH RISK PORTFOLIO

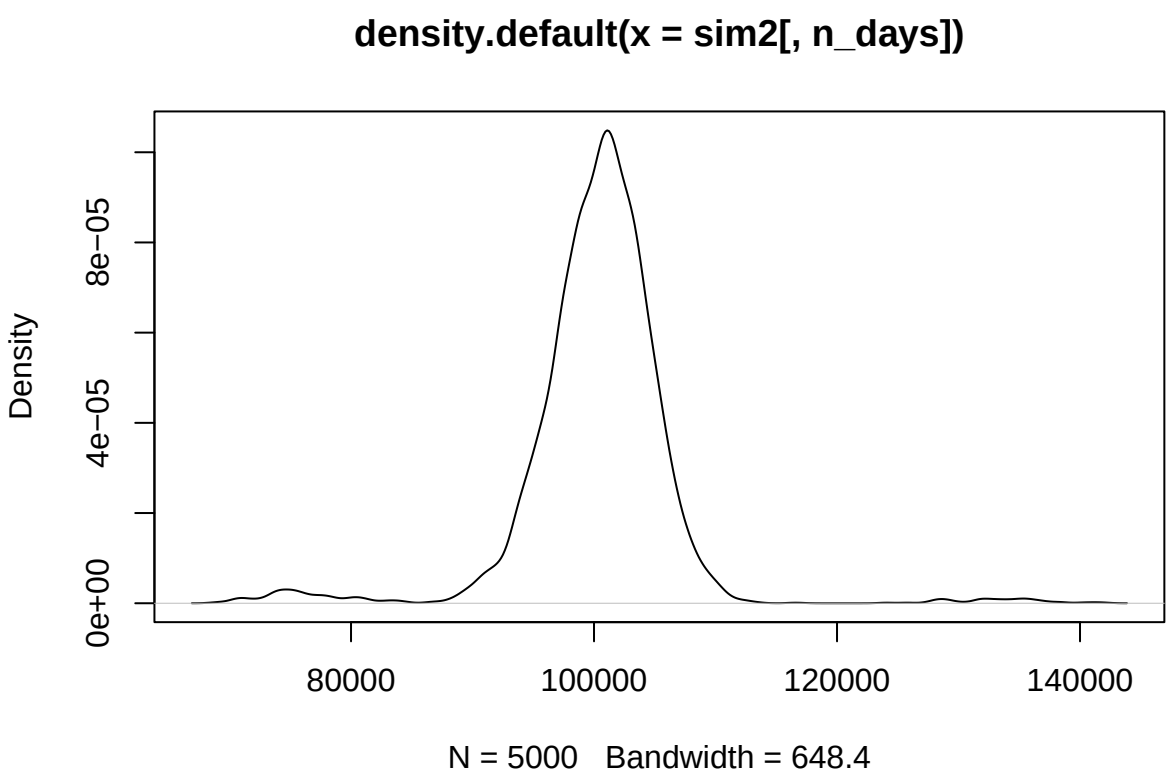
For the high risk portfolio, we distributed 90% of the total wealth among the low performing ETFs - SVXY, EPV and YYY.

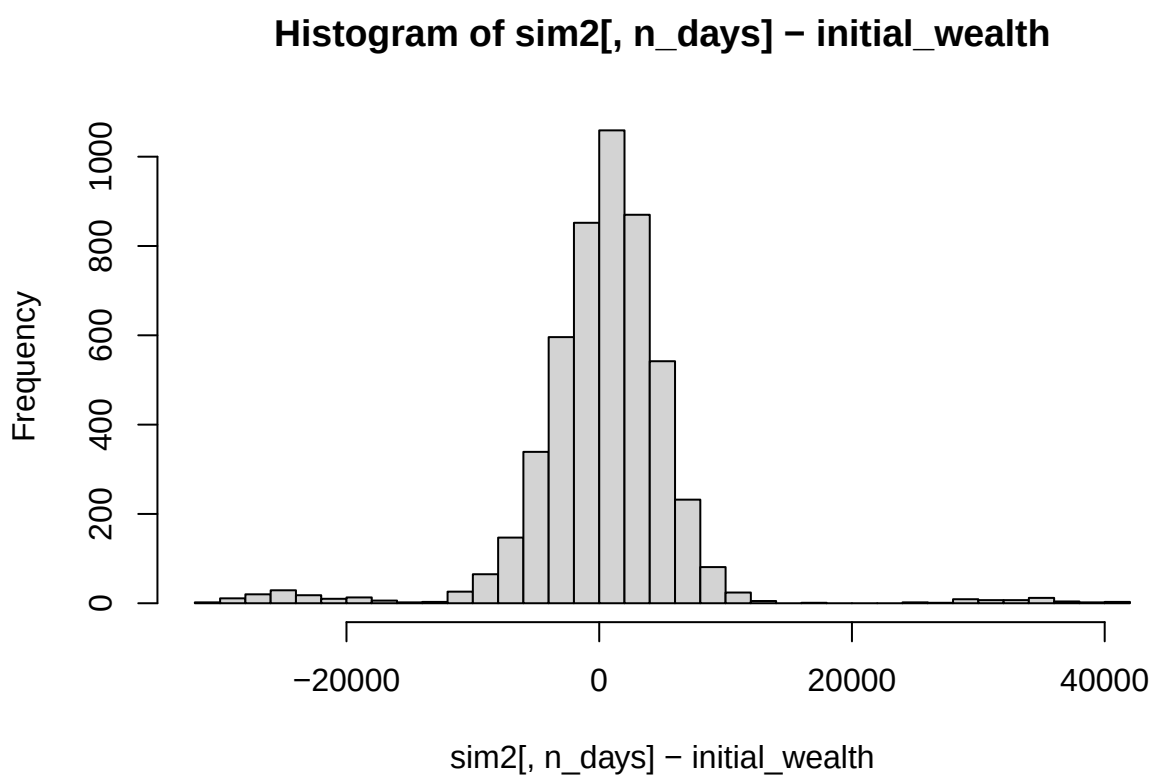
Average return of investement after 20 days - $100724.3
5% Value at Risk for safe portfolio - $7850.127

```
##              [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## result.1 101062.55  99709.65  99387.46  99816.19  97685.30  98214.91  99031.80
## result.2  99663.98 100041.24  99988.88 100012.43  98714.67  98739.08  97743.90
## result.3  99934.72  99749.71  99537.58 101289.32 101350.00 101047.50 102007.79
## result.4  99983.67  99610.73  99721.29  99923.08 100190.56 100975.98 101648.62
## result.5 100402.23 100081.84 100851.42 101420.59 102091.45 101621.49 102071.55
## result.6 100198.17  98761.01  98751.48  97851.99  98934.39  98964.61  99095.06
##              [,8]      [,9]     [,10]     [,11]     [,12]     [,13]     [,14]
## result.1  99244.05  99744.78  98394.60  98656.47  98250.69  98734.22  99083.07
## result.2  98016.12  98861.99  98950.83  99041.20  98962.60  99049.52  99563.25
```

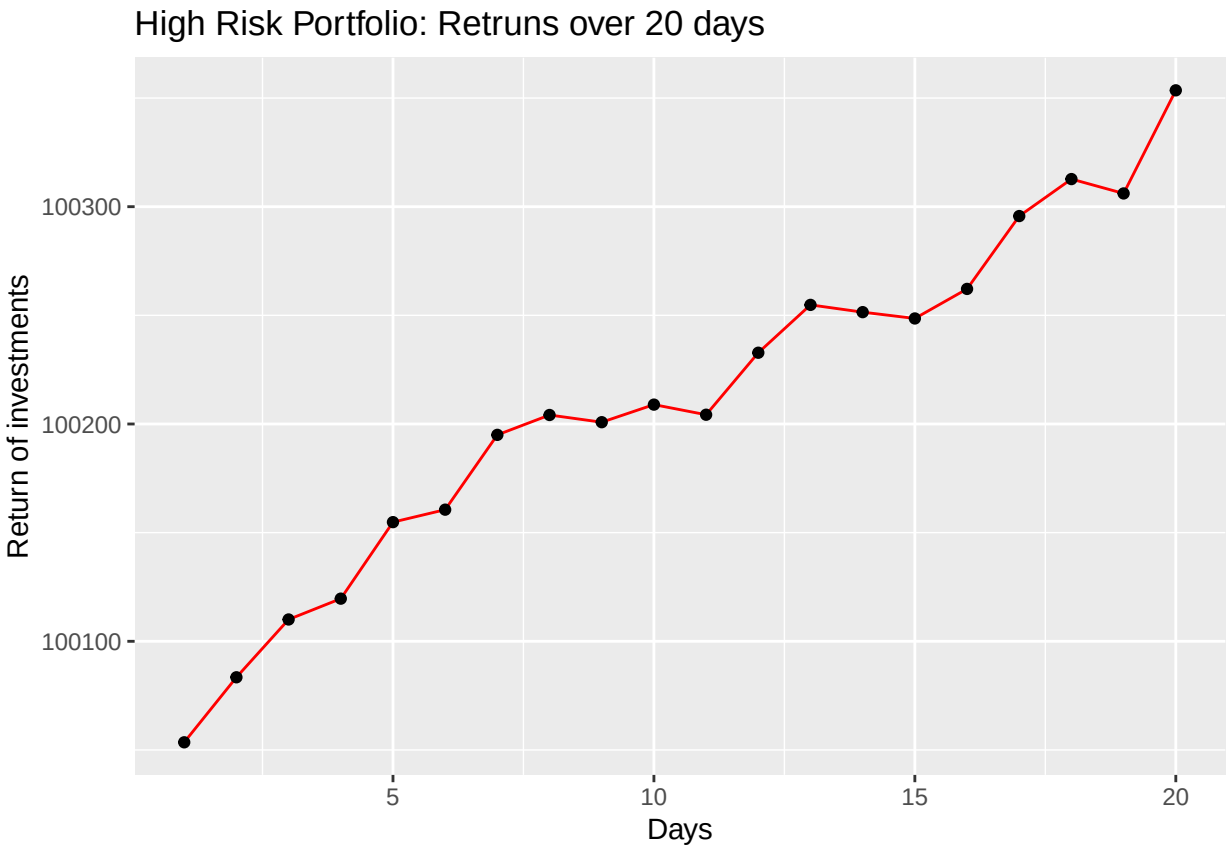
```
## result.3 102358.91 102494.29 101774.06 101685.10 100905.42  99631.05  99556.71
## result.4 101262.34 101123.89 100737.60 101264.94 101353.16 101594.74 101567.29
## result.5 102475.89 102434.12 103891.62 105382.11 105641.18 105272.50 105547.09
## result.6  99273.33  98509.06  98164.87  97670.34  98961.18  98554.15  96887.67
##             [,15]     [,16]     [,17]     [,18]     [,19]     [,20]
## result.1  97954.92  99290.89 101320.36 100567.94 101155.51 100814.90
## result.2  99078.90  99963.97 100689.85 100388.42 100855.61 100749.07
## result.3  99922.12  99840.30 100265.24  99863.49 100180.59 100781.92
## result.4 102443.81 102203.06 101146.56 101398.74 101782.57 101845.16
## result.5 105556.08 105038.24 105695.47 105573.56 104758.13 104781.00
## result.6  96523.05  96775.76  97446.38  97275.98  96900.39  96526.25
```

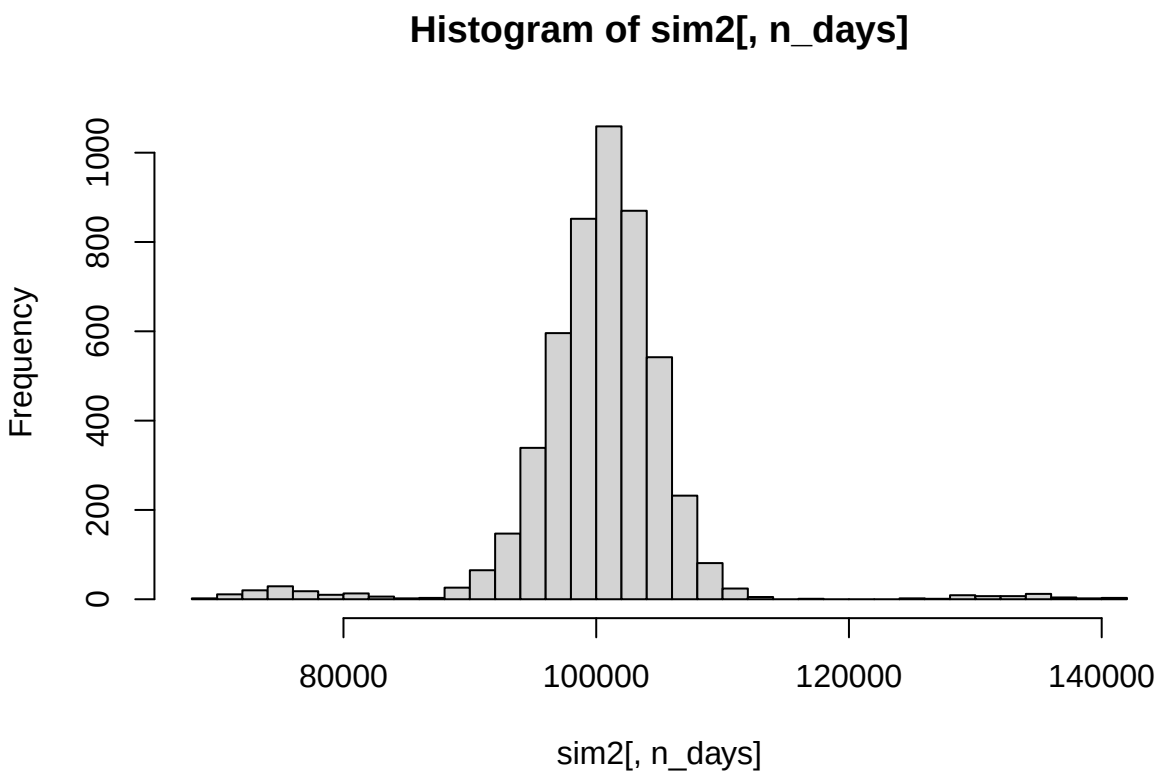


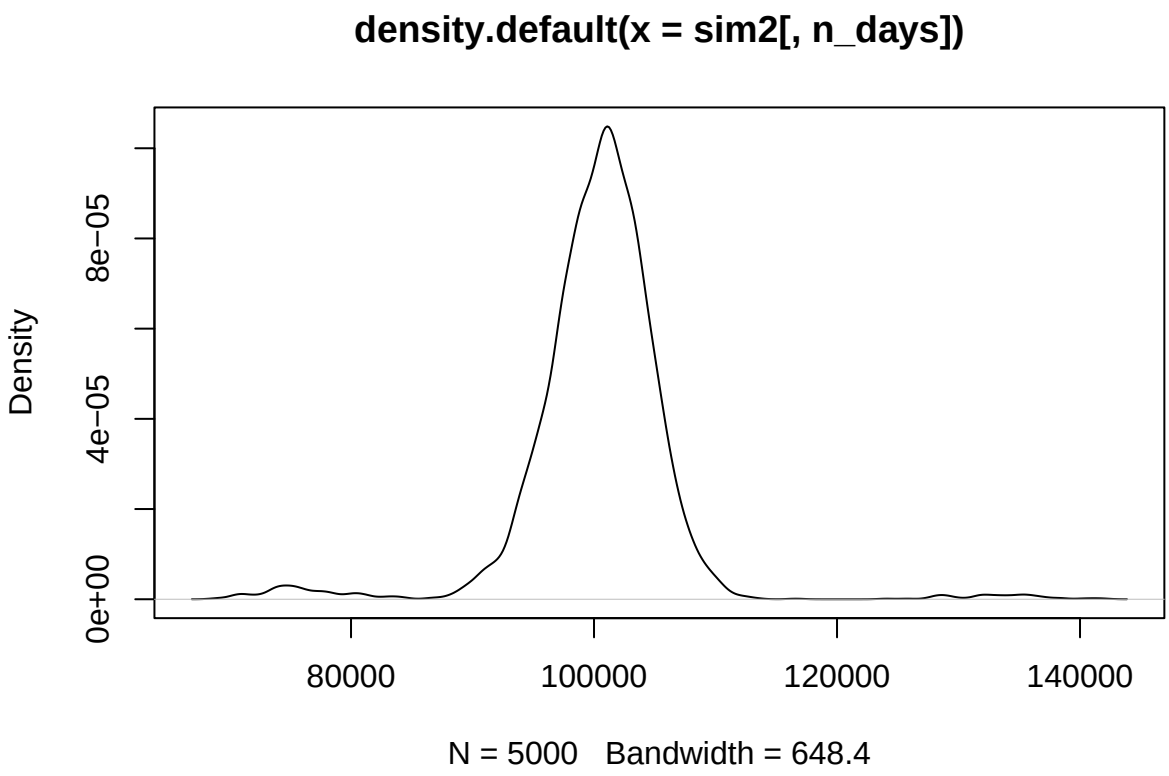

density.default(x = sim2[, n_days])
Density
0e+00
4e−05
8e−05
80000
100000
120000
140000
N = 5000 Bandwidth = 648.4

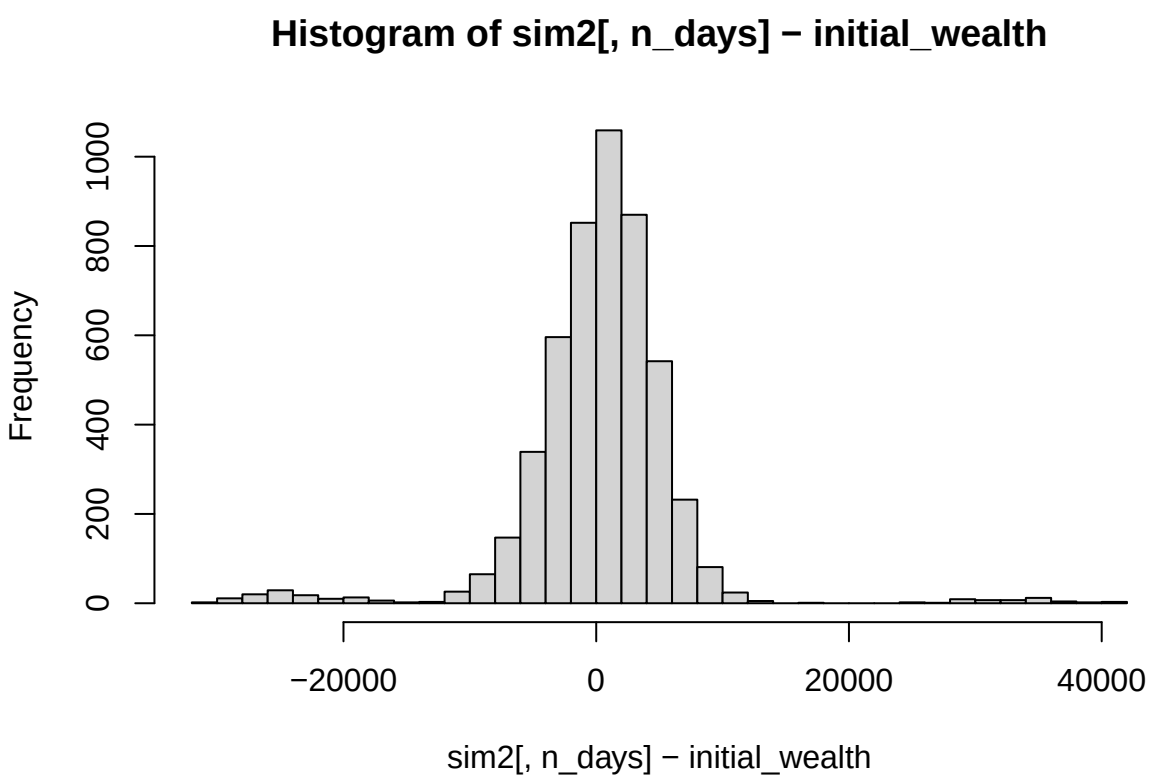



```
## Confidence Interval from Bootstrap Distribution (5000 replicates)

##
## Average return of investement after 20 days 100353.5

##
## 5% Value at Risk for High portfolio- -6968.49
```


High Risk Portfolio: Retruns over 20 days
Return of investments
100300
100200
100100
5
10
15
20
Days


Histogram of sim2[, n_days]
Frequency
0
200
400
600
800
1000
80000
100000
120000
140000
sim2[, n_days]


density.default(x = sim2[, n_days])
Density
0e+00
4e−05
8e−05
80000
100000
120000
140000
N = 5000 Bandwidth = 648.4

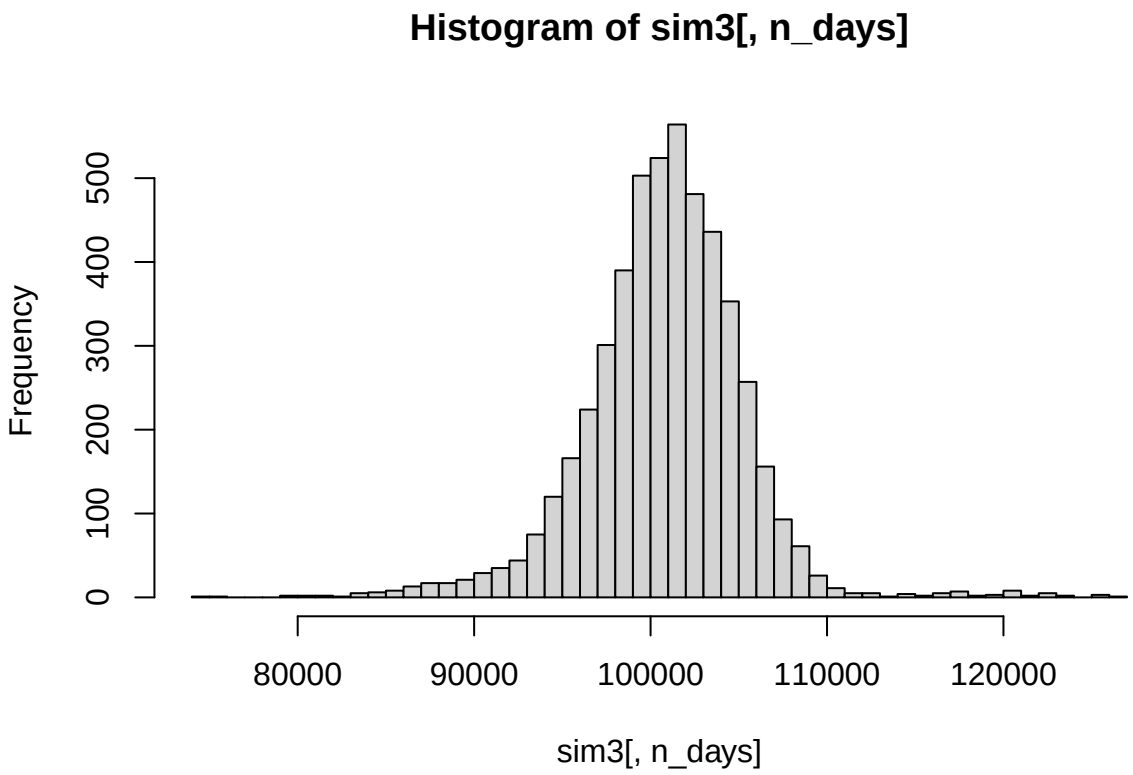


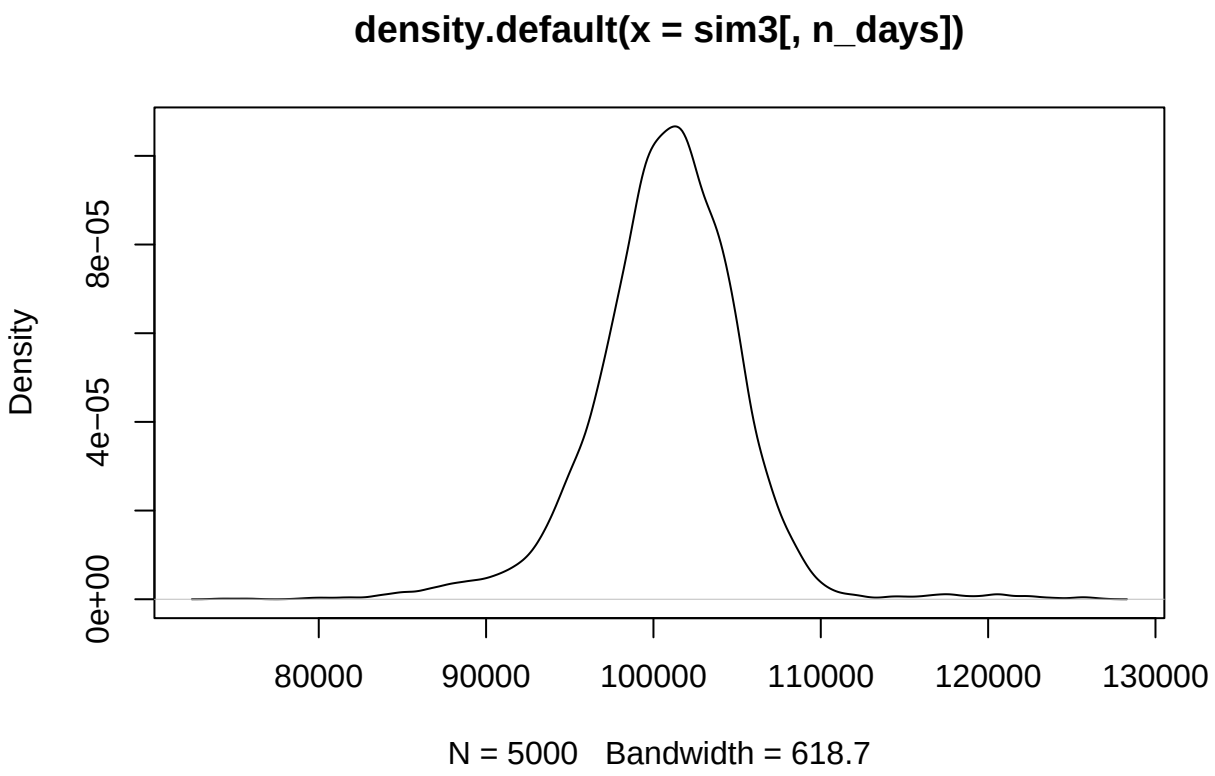

```
## Confidence Interval from Bootstrap Distribution (5000 replicates)

##
## Average return of investement after 20 days 100353.5

##
## 5% Value at Risk for High portfolio- -6968.49
```

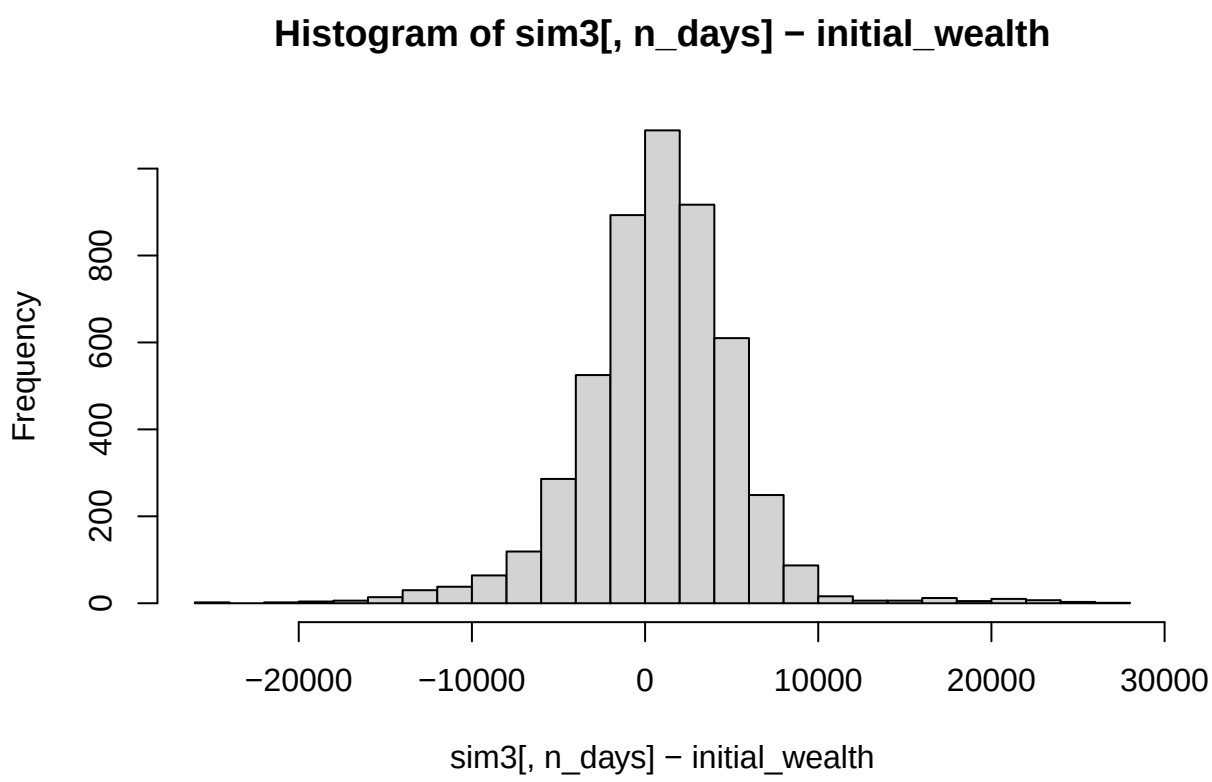
SIMULATION 3 - With equal weights for high risk and low risk

```
##              [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## result.1  99565.09  99482.13  99558.83  99242.56  99873.09  99221.99  99475.15
## result.2  99959.11 100461.65  99691.94  99836.36  98838.89  99085.82  99158.75
## result.3 100351.53 100889.36 100776.19  97951.49  99412.03  99725.14  99925.01
## result.4  99744.35 101795.13 101615.28 101958.12 100798.32 100620.48 100456.09
## result.5  99210.51  99670.42  99540.40  96434.24  97263.33  97076.23  96170.89
## result.6  98799.98  99078.70 100177.38 100450.93 100462.38 100722.99 101089.61
##              [,8]      [,9]     [,10]     [,11]     [,12]     [,13]     [,14]
## result.1  99319.59 100493.73 100700.03 100439.86  99801.27  98601.10  99923.57
## result.2  99409.98  99955.42  99187.24  99526.66  98758.90 100702.40 100407.39
## result.3  99683.08  99984.48 100311.37 100317.25 100393.95 100677.17  99574.45
## result.4  99733.85  99419.30  99387.23 100361.80 100200.66  99373.19  99568.88
## result.5  95580.11  95786.21  95762.53  95951.11  96603.56  96010.68  96103.15
## result.6 100532.05 100991.95 102114.75 102274.71 101937.98 102109.61 102309.71
##             [,15]     [,16]     [,17]     [,18]     [,19]     [,20]
```

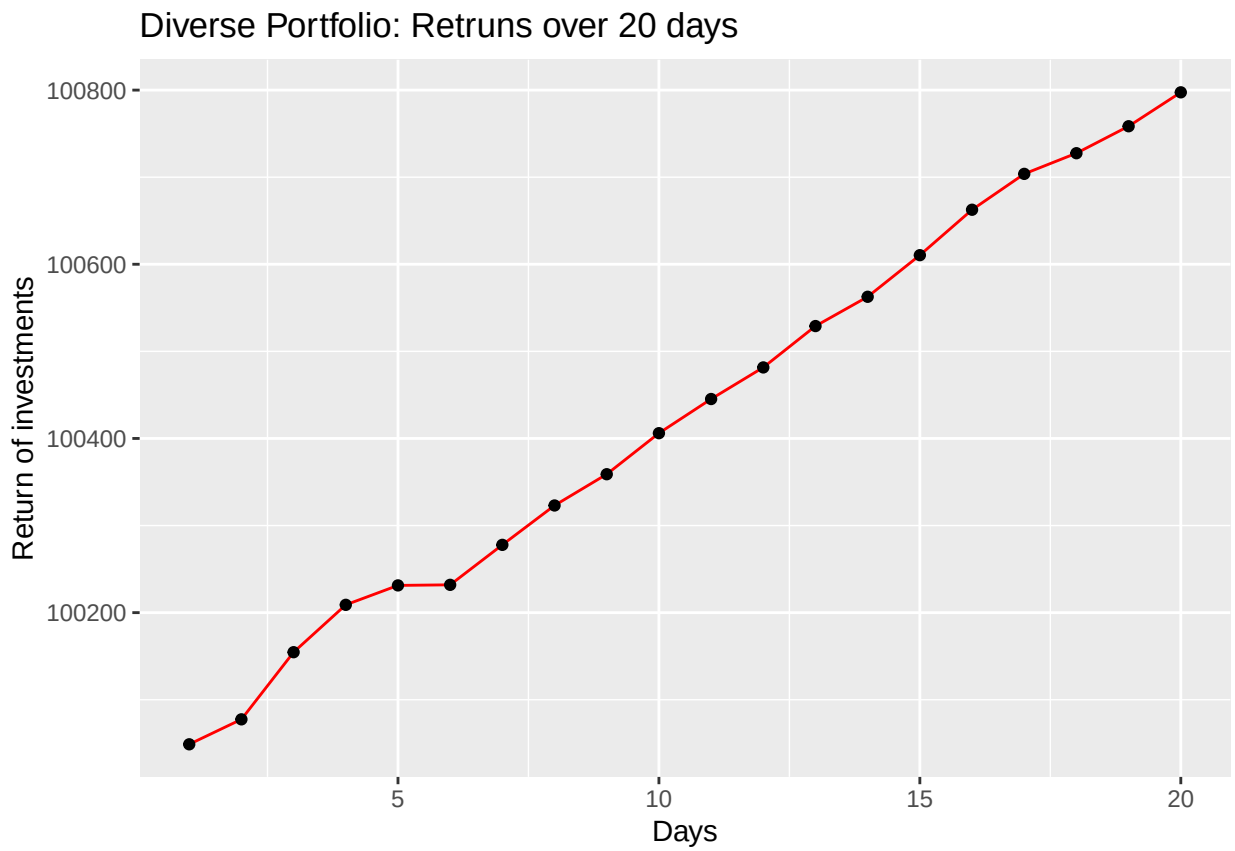
```
## result.1  99496.36  99469.70 100461.54 100045.87  99420.03  98598.45
## result.2 100314.95 100725.88 100971.08 101005.94 101198.57 101144.33
## result.3  99694.79 100350.03  99093.67  99126.32  96829.99  97203.94
## result.4 100383.22 100736.41 100208.45 100377.21 100610.81 100769.67
## result.5  97110.70  97251.53  98128.72  97787.72  97535.08  97165.69
## result.6 101971.04 100870.69 100673.21  99619.84  97689.36  97689.34
```




density.default(x = sim3[, n_days])
Density
0e+00
4e−05
8e−05
80000
90000
100000
110000
120000
130000
N = 5000 Bandwidth = 618.7




```
## Confidence Interval from Bootstrap Distribution (5000 replicates)

##
## Average return of investement after 20 days 100797.4

##
## 5% Value at Risk for High portfolio- -6375.387
```



Summary

For the safe portfolio, we are observing the maximum return of investment and the lowest 5% VaR. As the portfolio risk increases, we are able to witness the decrease in returns and increase in VaR value as expected.

References: https://www.bankrate.com/investing/best-etfs/ https://etfdb.com/compare/lowest-ytd-returns/

Problem 4: Market Segmentation

Market Segmentation

```
## corrplot 0.90 loaded
```

Data import and Exploration

```
##           X chatter current_events travel photo_sharing uncategorized tv_film
## 1 hmjoe4g3k       2              0      2             2             2       1
## 2 clk1m5w8s       3              3      2             1             1       1
## 3 jcsovtak3       6              3      4             3             1       5
## 4 3oeb4hiln       1              5      2             2             0       1
## 5 fd75x1vgk       5              2      0             6             1       0
## 6 h6nvj91yp       6              4      2             7             0       1
##   sports_fandom politics food family home_and_garden music news online_gaming
```

```
## 1            1        0    4      1              2     0    0            0
## 2            4        1    2      2              1     0    0            0
## 3            0        2    1      1              1     1    1            0
## 4            0        1    0      1              0     0    0            0
## 5            0        2    0      1              0     0    0            3
## 6            1        0    2      1              1     1    0            0
##   shopping health_nutrition college_uni sports_playing cooking eco computers
## 1        1               17           0              2       5   1         1
## 2        0                0           0              1       0   0         0
## 3        2                0           0              0       2   1         0
## 4        0                0           1              0       0   0         0
## 5        2                0           4              0       1   0         1
## 6        5                0           0              0       0   0         1
##   business outdoors crafts automotive art religion beauty parenting dating
## 1        0        2      1          0   0        1      0         1      1
## 2        1        0      2          0   0        0      0         0      1
## 3        0        0      2          0   8        0      1         0      1
## 4        1        0      3          0   2        0      1         0      0
## 5        0        1      0          0   0        0      0         0      0
## 6        1        0      0          1   0        0      0         0      0
##   school personal_fitness fashion small_business spam adult
## 1      0               11       0              0    0     0
## 2      4                0       0              0    0     0
## 3      0                0       1              0    0     0
## 4      0                0       0              0    0     0
## 5      0                0       0              1    0     0
## 6      0                0       0              0    0     0

##       X                chatter       current_events      travel
##  Length:7882        Min.   : 0.000   Min.   :0.000   Min.   : 0.000
##  Class :character   1st Qu.: 2.000   1st Qu.:1.000   1st Qu.: 0.000
##  Mode  :character   Median : 3.000   Median :1.000   Median : 1.000
##                     Mean   : 4.399   Mean   :1.526   Mean   : 1.585
##                     3rd Qu.: 6.000   3rd Qu.:2.000   3rd Qu.: 2.000
##                     Max.   :26.000   Max.   :8.000   Max.   :26.000
##  photo_sharing    uncategorized      tv_film      sports_fandom
##  Min.   : 0.000   Min.   :0.000   Min.   : 0.00   Min.   : 0.000
##  1st Qu.: 1.000   1st Qu.:0.000   1st Qu.: 0.00   1st Qu.: 0.000
##  Median : 2.000   Median :1.000   Median : 1.00   Median : 1.000
##  Mean   : 2.697   Mean   :0.813   Mean   : 1.07   Mean   : 1.594
##  3rd Qu.: 4.000   3rd Qu.:1.000   3rd Qu.: 1.00   3rd Qu.: 2.000
##  Max.   :21.000   Max.   :9.000   Max.   :17.00   Max.   :20.000
##     politics           food            family        home_and_garden
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.0000   Min.   :0.0000
##  1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.:0.0000
##  Median : 1.000   Median : 1.000   Median : 1.0000   Median :0.0000
##  Mean   : 1.789   Mean   : 1.397   Mean   : 0.8639   Mean   :0.5207
##  3rd Qu.: 2.000   3rd Qu.: 2.000   3rd Qu.: 1.0000   3rd Qu.:1.0000
##  Max.   :37.000   Max.   :16.000   Max.   :10.0000   Max.   :5.0000
##      music              news        online_gaming       shopping
##  Min.   : 0.0000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 0.0000   1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000
##  Median : 0.0000   Median : 0.000   Median : 0.000   Median : 1.000
##  Mean   : 0.6793   Mean   : 1.206   Mean   : 1.209   Mean   : 1.389
```

```
##  3rd Qu.: 1.0000   3rd Qu.: 1.000   3rd Qu.: 1.000   3rd Qu.: 2.000
##  Max.   :13.0000   Max.   :20.000   Max.   :27.000   Max.   :12.000
##  health_nutrition  college_uni     sports_playing      cooking
##  Min.   : 0.000   Min.   : 0.000   Min.   :0.0000   Min.   : 0.000
##  1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.:0.0000   1st Qu.: 0.000
##  Median : 1.000   Median : 1.000   Median :0.0000   Median : 1.000
##  Mean   : 2.567   Mean   : 1.549   Mean   :0.6392   Mean   : 1.998
##  3rd Qu.: 3.000   3rd Qu.: 2.000   3rd Qu.:1.0000   3rd Qu.: 2.000
##  Max.   :41.000   Max.   :30.000   Max.   :8.0000   Max.   :33.000
##       eco            computers          business         outdoors
##  Min.   :0.0000   Min.   : 0.0000   Min.   :0.0000   Min.   : 0.0000
##  1st Qu.:0.0000   1st Qu.: 0.0000   1st Qu.:0.0000   1st Qu.: 0.0000
##  Median :0.0000   Median : 0.0000   Median :0.0000   Median : 0.0000
##  Mean   :0.5123   Mean   : 0.6491   Mean   :0.4232   Mean   : 0.7827
##  3rd Qu.:1.0000   3rd Qu.: 1.0000   3rd Qu.:1.0000   3rd Qu.: 1.0000
##  Max.   :6.0000   Max.   :16.0000   Max.   :6.0000   Max.   :12.0000
##      crafts          automotive           art             religion
##  Min.   :0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.000
##  1st Qu.:0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.000
##  Median :0.0000   Median : 0.0000   Median : 0.0000   Median : 0.000
##  Mean   :0.5159   Mean   : 0.8299   Mean   : 0.7248   Mean   : 1.095
##  3rd Qu.:1.0000   3rd Qu.: 1.0000   3rd Qu.: 1.0000   3rd Qu.: 1.000
##  Max.   :7.0000   Max.   :13.0000   Max.   :18.0000   Max.   :20.000
##      beauty          parenting           dating            school
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.7052   Mean   : 0.9213   Mean   : 0.7109   Mean   : 0.7677
##  3rd Qu.: 1.0000   3rd Qu.: 1.0000   3rd Qu.: 1.0000   3rd Qu.: 1.0000
##  Max.   :14.0000   Max.   :14.0000   Max.   :24.0000   Max.   :11.0000
##  personal_fitness    fashion        small_business        spam
##  Min.   : 0.000   Min.   : 0.0000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.:0.0000   1st Qu.:0.00000
##  Median : 0.000   Median : 0.0000   Median :0.0000   Median :0.00000
##  Mean   : 1.462   Mean   : 0.9966   Mean   :0.3363   Mean   :0.00647
##  3rd Qu.: 2.000   3rd Qu.: 1.0000   3rd Qu.:1.0000   3rd Qu.:0.00000
##  Max.   :19.000   Max.   :18.0000   Max.   :6.0000   Max.   :2.00000
##      adult
##  Min.   : 0.0000
##  1st Qu.: 0.0000
##  Median : 0.0000
##  Mean   : 0.4033
##  3rd Qu.: 0.0000
##  Max.   :26.0000
```

```
##  [1] "X"                "chatter"          "current_events"   "travel"
##  [5] "photo_sharing"    "uncategorized"    "tv_film"          "sports_fandom"
##  [9] "politics"         "food"             "family"           "home_and_garden"
## [13] "music"            "news"             "online_gaming"    "shopping"
## [17] "health_nutrition" "college_uni"      "sports_playing"   "cooking"
## [21] "eco"              "computers"        "business"         "outdoors"
## [25] "crafts"           "automotive"       "art"              "religion"
## [29] "beauty"           "parenting"        "dating"           "school"
## [33] "personal_fitness" "fashion"          "small_business"   "spam"
```

```
## [37] "adult"
```

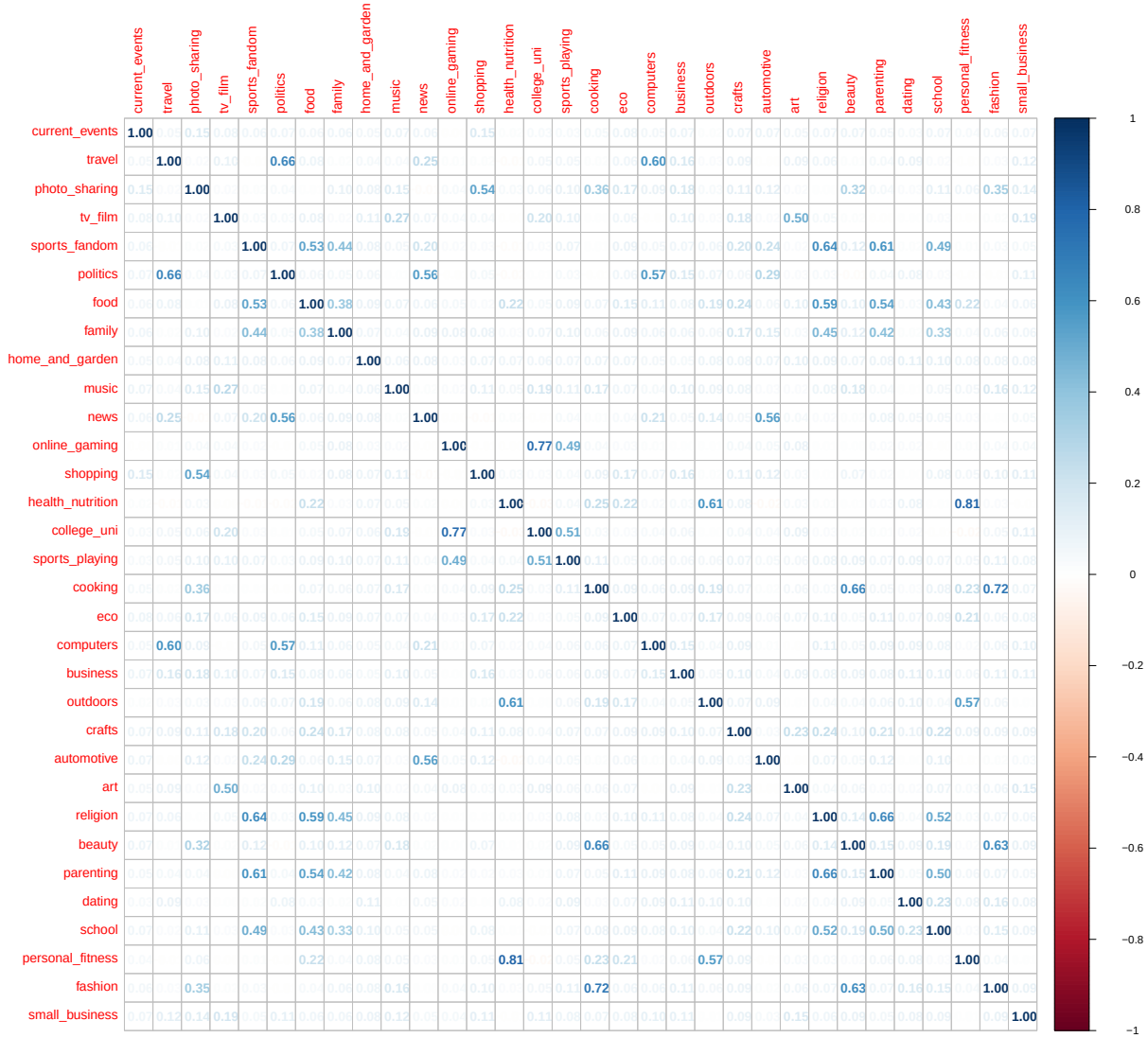
To perform cluster analysis on the data, I will be using K-Means Clustering approach. Following columns shall be removed to perform cluster analysis : * X : Since it is unique value for each user, doesn't make much sense to keep it * *chatter* : Random values which doesn't fit in any column. It won't lie in any segment. * *adult* : Target variable, no need in cluster analysis * *spam* : Target variable

Data Standardization

```
## Scaled Mean :
##  1.526262 1.585004 2.696777 1.070287 1.594012 1.788632 1.397488 0.863867 0.52068 0.6792692 1.205532 1
```

```
## Scaled SD :
##  1.26889 2.28553 2.73151 1.658783 2.160917 3.031113 1.775557 1.132562 0.7366913 1.030015 2.10078 2.68
```

```
corrplot(cor(mkt_km), method = "number")
```




```
##                Var1               Var2      corr
## 1  personal_fitness health_nutrition 0.8099024
## 2       college_uni    online_gaming 0.7728393
## 3           fashion          cooking 0.7214027
## 4            beauty          cooking 0.6642389
## 5          politics           travel 0.6602100
## 6         parenting         religion 0.6555973
## 7          religion    sports_fandom 0.6379748
## 8           fashion           beauty 0.6349739
## 9          outdoors health_nutrition 0.6082254
## 10        parenting    sports_fandom 0.6077181
```

Let's see if there is any highly negatively correlated variables ?

```
##                 Var1             Var2         corr
## 487 personal_fitness       automotive -0.009861229
## 488 health_nutrition    sports_fandom -0.011229255
## 489           beauty         politics -0.011292710
## 490         shopping             news -0.011813142
## 491 health_nutrition           travel -0.011922499
## 492             news    photo_sharing -0.011980028
## 493 health_nutrition         politics -0.016851900
## 494 personal_fitness      college_uni -0.021526868
## 495       automotive health_nutrition -0.023824999
## 496      college_uni health_nutrition -0.027778856
```

The data has almost positive correlation with personal_fitness and health_nutrition being top correlated variables.

Clustering

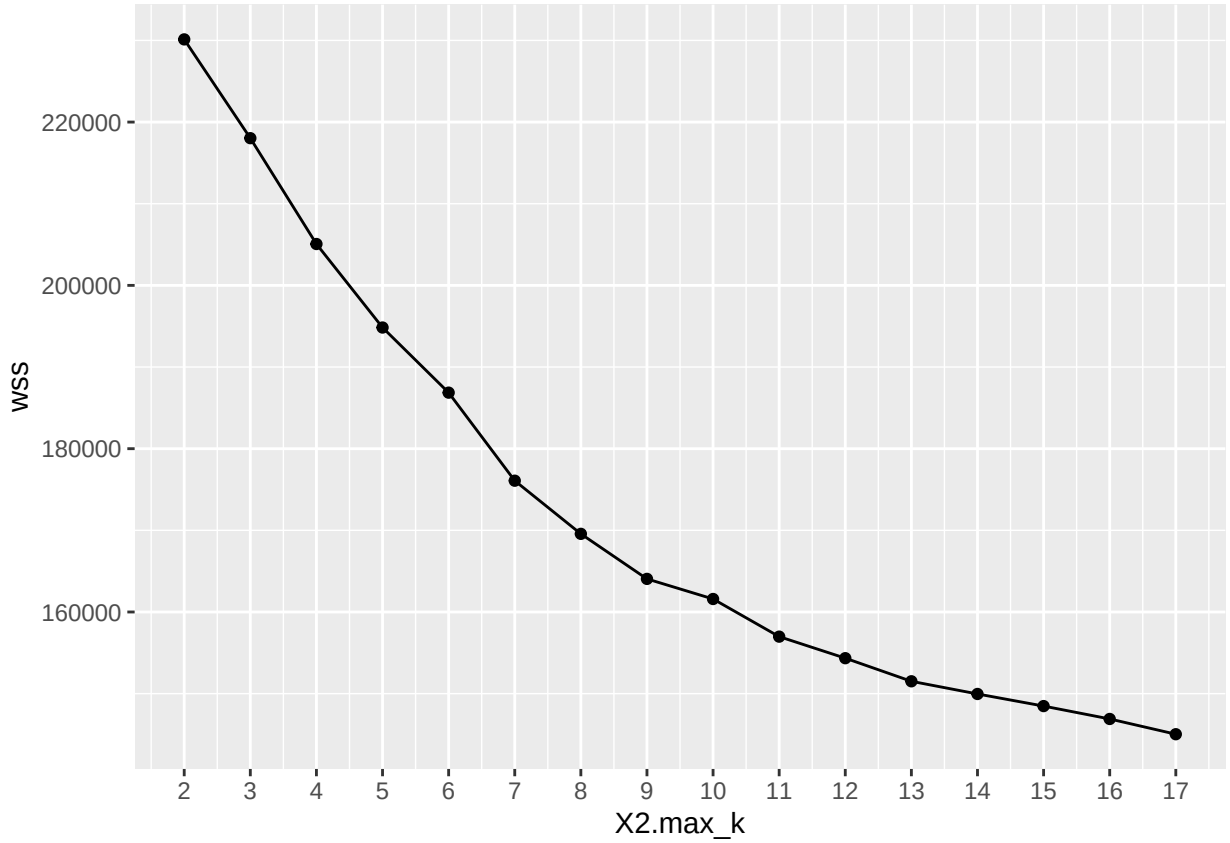
Distribution of columns in each cluster :

```
## cluster No : 1
## health_nutrition personal_fitness         outdoors              eco
##        1723.5995        1673.6716        1337.3631         444.8355
##             food
##         358.1745
##
## cluster No : 2
##      religion     parenting sports_fandom          food        school
##      1537.241      1453.985      1402.637      1241.678      1116.322
##
## cluster No : 3
##    travel  politics computers      news  business
## 1150.7524 1094.7797 1026.3450  403.0183  193.5624
##
## cluster No : 4
##          news    automotive      politics sports_fandom      outdoors
##     1129.1330     1109.6628      522.3851      285.8661      130.5772
##
## cluster No : 5
##          dating          school         fashion home_and_garden        business
##       961.48250       257.65186       165.00401       123.93517        93.99723
##
## cluster No : 6
## home_and_garden          dating          travel  small_business         tv_film
##       -697.6195       -717.1647       -717.4607       -729.4961       -732.8135
##
## cluster No : 7
##       shopping  photo_sharing            eco       business small_business
##      1383.2380      1083.5140       313.5060       308.0282       180.0068
##
## cluster No : 8
##  online_gaming    college_uni sports_playing            art         family
##     1280.84978     1176.42546      761.82617      100.34261       74.81764
##
```

```
## cluster No : 9
##       cooking       fashion        beauty photo_sharing         music
##     1351.8754     1291.6365     1262.5378      606.4583      261.1135
##
## cluster No : 10
##        tv_film            art          music small_business         crafts
##      1119.9018      1087.3378       408.0082       340.4638       312.3090
```

FInding the optimal value of k for K-Means Clustering

```
## [1] 230126.3
```




```
Optimal K : 10
```

```
##  [1]  407 3130  357  480  793  334  427  744  852  358
```

```
##    current_events      travel photo_sharing     tv_film sports_fandom
## 1     0.319130438  0.22135146   -0.05989652  2.78673404    -0.1248022
## 2    -0.212809814 -0.20982993   -0.41135289 -0.21687196    -0.3946461
## 3     0.110651237  3.24799987   -0.10331708 -0.06263638    -0.1971128
## 4     0.191102102 -0.05451128    1.27184447 -0.13907989    -0.2208992
## 5    -0.004299274 -0.15388666   -0.08842881 -0.14576187    -0.2077788
## 6     0.255370378  0.02568716    0.10881692  0.01358082     2.8217667
## 7     0.052205835 -0.18833144   -0.21479226 -0.01413586     0.6636493
## 8    -0.027052679 -0.24419815   -0.24819981 -0.21982471     0.8926013
```

```
## 9     0.415897405 -0.20820071    1.23508429 -0.14426285    -0.2488173
## 10   -0.093342291 -0.03841434   -0.01068263  0.10413056    -0.1417464
##       politics       food      family home_and_garden       music        news
## 1  -0.08995434  0.1248491 -0.12494534      0.30044433  1.05563103  0.03081649
## 2  -0.29538372 -0.4343931 -0.35738588     -0.21151846 -0.23142838 -0.30618685
## 3   3.08884119  0.1594904 -0.08260822      0.03086413 -0.04214976  1.12619921
## 4  -0.12546511 -0.2144799  0.03374323      0.13595017  0.55005436 -0.06510990
## 5  -0.19403032  0.4472901 -0.08912665      0.15594203  0.01388282 -0.07502560
## 6  -0.13078255  2.5483085  2.03679093      0.31331575  0.21255460  0.01617940
## 7   1.23562695 -0.1618746  0.21325050      0.16107729 -0.08650875  2.67017785
## 8  -0.32447640  0.7693094  0.62220122      0.06497750 -0.12576328 -0.29105643
## 9  -0.13007303 -0.3699557 -0.06012245      0.11372480  0.13248206 -0.26432887
## 10 -0.16526033 -0.1090234  0.19665606      0.05913709 -0.05200983 -0.19357042
##    online_gaming    shopping health_nutrition   college_uni sports_playing
## 1    -0.16547285  0.07542397     -0.156217630  0.3904209751     0.10541782
## 2    -0.23177522 -0.39400308     -0.310984756 -0.2542840899    -0.25729842
## 3    -0.15587712 -0.05110961     -0.161673134 -0.0349655355     0.03105012
## 4    -0.01878979  0.21319219     -0.050634162 -0.0106069001     0.20329952
## 5    -0.11665123 -0.02842202      2.200618295 -0.2070725625    -0.01534163
## 6     0.04595598  0.11412813     -0.008962282 -0.0005438564     0.24711170
## 7    -0.13086213 -0.18936371     -0.249608766 -0.1977454248    -0.09345661
## 8    -0.18623384 -0.18776447     -0.292074625 -0.2295601577    -0.08066903
## 9    -0.17684405  1.60677395     -0.273653906 -0.1199804508    -0.08852842
## 10    3.57467454 -0.12723599     -0.184560739  3.2861046388     2.11941585
##       cooking          eco   computers     business    outdoors      crafts
## 1  -0.1427501  0.119674038 -0.14826561  0.385879509 -0.08439945  0.75211369
## 2  -0.3256831 -0.292008996 -0.25885203 -0.255058964 -0.32484379 -0.32253602
## 3  -0.1864990  0.164148187  2.93845645  0.541721383 -0.03337993  0.20862595
## 4   2.8109405  0.011077740  0.07850487  0.213164332  0.02294145  0.08515389
## 5   0.4089823  0.559871411 -0.08522934  0.066238051  1.71964222  0.05391483
## 6   0.0991561  0.435218976  0.23914799  0.240587212  0.03116200  0.97388838
## 7  -0.2439200 -0.099658118 -0.19290986 -0.147895508  0.31716609 -0.15273134
## 8  -0.2820194 -0.009003447 -0.12069440  0.002149619 -0.19703836  0.33271263
## 9  -0.2383343  0.337775171 -0.03185501  0.370223017 -0.29969962  0.03806027
## 10 -0.1273408 -0.074051471 -0.08139467 -0.115063304 -0.15518806  0.01478127
##     automotive          art     religion       beauty   parenting      dating
## 1  -0.20458718  2.647619458 -0.003632265  0.001856861 -0.19940835 -0.06661026
## 2  -0.31722694 -0.238536835 -0.384859752 -0.286339370 -0.38662567 -0.17925039
## 3  -0.13586186 -0.162884304  0.134494440 -0.189298091  0.03157531  0.32724126
## 4   0.02998777  0.009062051 -0.158625398  2.634993845 -0.08007330  0.04299938
## 5  -0.16715010 -0.067153327 -0.172976563 -0.206248028 -0.09455304  0.17778949
## 6   0.31520201  0.088027483  3.032014275  0.578282476  2.98593893 -0.01583612
## 7   2.59305153 -0.168858277 -0.190475498 -0.180065004  0.03336319 -0.05195016
## 8  -0.11257193 -0.122288530  1.064145539  0.067176500  0.89985733  0.61996910
## 9   0.10735293 -0.220789123 -0.326269249 -0.262326028 -0.27804625 -0.13806036
## 10  0.06319624  0.273430108 -0.205913864 -0.226012812 -0.14164031 -0.01800392
##         school personal_fitness     fashion small_business
## 1  -0.03815652      -0.14920371 -0.02097093     0.82719813
## 2  -0.38553364      -0.33038364 -0.29867181    -0.21900540
## 3  -0.09916911      -0.14086560 -0.17890203     0.38484989
## 4   0.16393851      -0.02320574  2.71027961     0.18387872
## 5  -0.17275471       2.14358329 -0.10157970    -0.12589885
## 6   2.31705300       0.08547956  0.18527268     0.19211272
## 7   0.01220512      -0.23592278 -0.22739849    -0.15008770
```

```
## 8   0.93413089      -0.29381180 -0.02973627     0.01688634
## 9  -0.09589043      -0.21797040 -0.15283112     0.20400551
## 10 -0.21353260      -0.18746217 -0.07145690     0.10208802
```

```
## cluster No : 1
##  [1] "health_nutrition" "personal_fitness" "outdoors"         "eco"
##  [5] "food"             "cooking"          "home_and_garden"  "crafts"
##  [9] "business"         "dating"
##
## cluster No : 2
##  [1] "religion"        "parenting"       "sports_fandom"   "food"
##  [5] "school"          "family"          "crafts"          "beauty"
##  [9] "eco"             "home_and_garden"
##
## cluster No : 3
##  [1] "travel"         "politics"       "computers"      "news"
##  [5] "business"       "small_business" "dating"         "crafts"
##  [9] "eco"            "food"
##
## cluster No : 4
##  [1] "news"            "automotive"      "politics"        "sports_fandom"
##  [5] "outdoors"        "family"          "home_and_garden" "parenting"
##  [9] "school"          "tv_film"
##
## cluster No : 5
##  [1] "dating"          "school"          "fashion"         "home_and_garden"
##  [5] "business"        "crafts"          "sports_playing"  "small_business"
##  [9] "beauty"          "eco"
##
## cluster No : 6
##  [1] "home_and_garden" "dating"          "travel"          "small_business"
##  [5] "tv_film"         "music"           "online_gaming"   "art"
##  [9] "computers"       "business"
##
## cluster No : 7
##  [1] "shopping"        "photo_sharing"   "eco"             "business"
##  [5] "small_business"  "music"           "home_and_garden" "automotive"
##  [9] "crafts"          "computers"
##
## cluster No : 8
##  [1] "online_gaming"   "college_uni"     "sports_playing"  "art"
##  [5] "family"          "tv_film"         "small_business"  "home_and_garden"
##  [9] "automotive"      "crafts"
##
## cluster No : 9
##  [1] "cooking"        "fashion"        "beauty"         "photo_sharing"
##  [5] "music"          "business"       "shopping"       "sports_playing"
##  [9] "small_business" "school"
##
## cluster No : 10
##  [1] "tv_film"         "art"             "music"           "small_business"
##  [5] "crafts"          "college_uni"     "business"        "home_and_garden"
##  [9] "travel"          "food"
```

Insights

Some clusters turned out to be meaningful and informative, below are the some categories which can be clubbed together :

- **Cluster 1** contains categories like `personal_fitness`, `nutrition_health` and `outdoors` can be taken posts related to fitness.
- **Cluster 2** contains categories like `parenting`, `food`, `school`, `sports`, `religion`, `crafts`. These can be considered as posts from educational institutions.

- **Cluster 3** contains categories like `travel`, `politics`, `computers`, `news` which clearly show that the posts belong to news.
- **Cluster 6** contains categories like `home_and_garden`, `dating`, `travel`, `small_business`, `tv_film`,`music` can be related lifestyle posts.
- **Cluster 8** contains categories like `online_gaming`, `college_uni`, `sports_playing` can be related college online ganing event.

Problem 5: Author attribution

Problem

Predicting the authorship of the articles in the C50test directory using a model trained using the c50train directory in the Reuters C50 Corpus. Describe the pre-processing and analysis pipeline in detail

Analysis

Step 1: Reading in the files from C50train and C50test directories

```
## Loading required package: NLP

##
## Attaching package: 'NLP'

## The following object is masked from 'package:ggplot2':
##
##     annotate

##
## Attaching package: 'tm'

## The following object is masked from 'package:mosaic':
##
##     inspect

##
## Attaching package: 'proxy'

## The following object is masked from 'package:Matrix':
##
##     as.matrix
```

```
## The following objects are masked from 'package:stats':
##
##     as.dist, dist

## The following object is masked from 'package:base':
##
##     as.matrix
```

We have imported all the authors files for training into `author_dirs_train`. Now we will clean these files.

```
# Rolling all directories together into a single corpus and getting Author names

#Train Data
file_list_train = NULL
labels = NULL

for(author in author_dirs_train)
{
  author_name = substring(author, first = 21)
  files_to_add = Sys.glob(paste0(author, '/*.txt'))
  file_list_train = append(file_list_train, files_to_add)
  labels = append(labels, rep(author_name, length(files_to_add)))
}

#Read files using wrapper function
all_docs_train = lapply(file_list_train, readerPlain)

#Getting rid of '.txt' from filename
names(all_docs_train) = file_list_train
names(all_docs_train) = sub('.txt', '', names(all_docs_train))


#Test Data
file_list_test = NULL
labels = NULL

for(author in author_dirs_test)
{
  author_name = substring(author, first = 29)
  files_to_add = Sys.glob(paste0(author, '/*.txt'))
  file_list_test = append(file_list_test, files_to_add)
  labels = append(labels, rep(author_name, length(files_to_add)))
}

#Read files using wrapper function
all_docs_test = lapply(file_list_test, readerPlain)

#Getting rid of '.txt' from filename
names(all_docs_test) = file_list_test
names(all_docs_test) = sub('.txt', '', names(all_docs_test))
```

`all_docs_train` has all the files. `author_name` has the name of the authors who wrote the corresponding files. `file_list_train` has the full path of all the files.

Step 2: Creating the Document Term Matrix and TF-IDF for training

```
library(dplyr)

# Creating training corpus and processing
my_corpus_train = Corpus(VectorSource(all_docs_train))

#Preprocessing
my_corpus_train = my_corpus_train %>%
                  tm_map(content_transformer(tolower))  %>%                          #transform to lowerc
                  tm_map(content_transformer(removeNumbers)) %>%                     #remove numbers
                  tm_map(content_transformer(removePunctuation)) %>%                 #remove punctuation
                  tm_map(content_transformer(stripWhitespace)) %>%                   #remove excess space
                  tm_map(content_transformer(removeWords), stopwords("SMART")) %>% #remove smrt stop-wo
                  tm_map(content_transformer(removeWords), stopwords("en"))        #remove en stop-word

#Create Document Term Matrix
DTM_train= DocumentTermMatrix(my_corpus_train)
DTM_train# some basic summary statistics

#Removing Sparse Terms
DTM_train = removeSparseTerms(DTM_train, 0.95)
DTM_train # now ~ 660 terms (versus ~32000 before)

#Create Weight TfIdf
tfidf_train = weightTfIdf(DTM_train)
```

Step 3: Creating the Document Term Matrix and TF-IDF for testing

NOTE: The words which aren't present in the training data are dropped here from the test DTM.

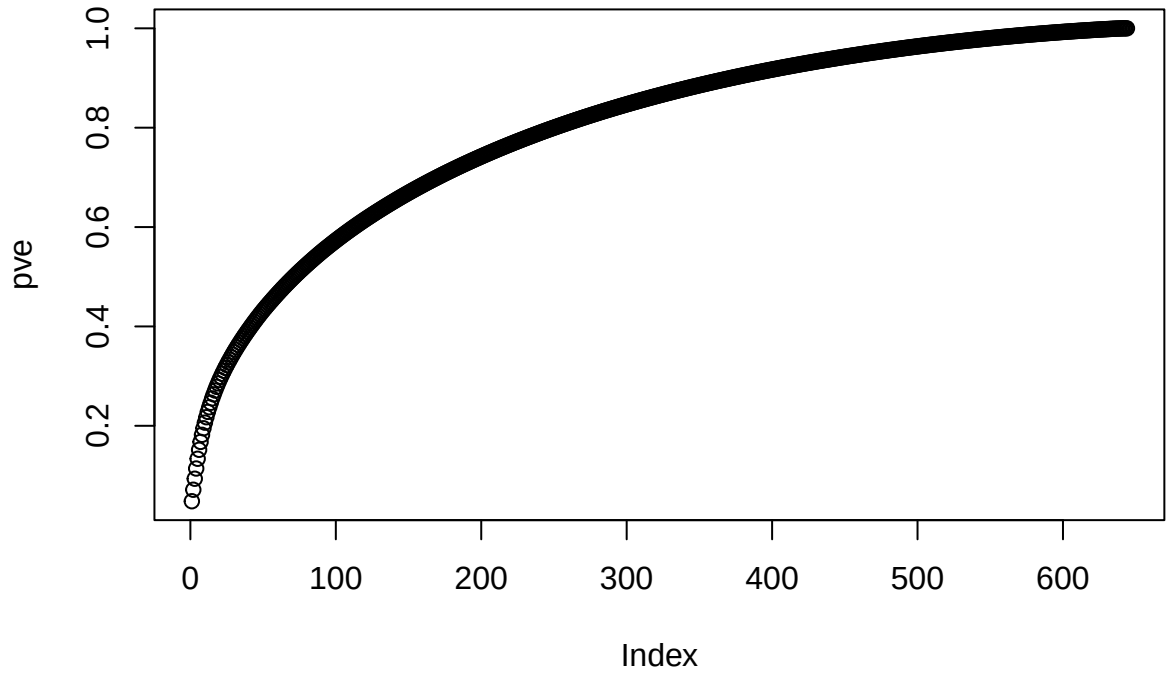
```
my_corpus_test = Corpus(VectorSource(all_docs_test))

#Pre-processing
my_corpus_test =  my_corpus_test %>%
                  tm_map(content_transformer(tolower))  %>%                          #transform to lowerc
                  tm_map(content_transformer(removeNumbers)) %>%                     #remove numbers
                  tm_map(content_transformer(removePunctuation)) %>%                 #remove punctuation
                  tm_map(content_transformer(stripWhitespace)) %>%                   #remove excess space
                  tm_map(content_transformer(removeWords), stopwords("SMART")) %>% #remove smrt stop-wo
                  tm_map(content_transformer(removeWords), stopwords("en"))        #remove en stop-word

#Create Document Term Matrix for Test - Specifying control to maintain same dim as train
DTM_test= DocumentTermMatrix(my_corpus_test, control = list(dictionary=Terms(DTM_train)))
DTM_test #Has the same 660 terms

#Create Weight TfIdf
tfidf_test = weightTfIdf(DTM_test)
```

`all_docs_train` has all the files. `author_name` has the name of the authors who wrote the corresponding files. `file_list_train` has the full path of all the files.

Step 4: Summarizing the Term matrices by using PCA

Dimensionality reduction

Principal component analysis is used to (1) extract relevant features from the huge set of variables (2) eliminate the effect of multi-collinearity while not losing out on relevant information from the correlated variables



Since we cant see much of an elbow, we will cut at 75 Here approximately 70 percent of the data is expalined

Step 5: Exploring Classification Models

```
set.seed(123)
n_cut = 75

#Create X & y using 75 as the cutoff for Train
X = pc_author$x[,1:n_cut]
y = sapply(strsplit(names(all_docs_train), "/"), "[", 4)

#Create X & y using 75 as the cutoff for Test
X_test = pc_author_test[,1:n_cut]
y_test = sapply(strsplit(names(all_docs_test), "/"), "[", 4)

#Combine X & y for Train & Validation
TrainSet <- cbind(as.data.frame(X),y)
ValidSet <- cbind(as.data.frame(X_test),y_test)
```

Model 1: Boosting

```
## Loaded gbm 2.1.8
```

```
## [1] 0.852
```

```
## [1] 0.404
```

Boosting classifies the training data well, but is unable to classify the test data.

Model 2: Random Forest

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
## [1] 1
```

```
## [1] 0.492
```

Random Forest does a better job than classification as compared to Boosting.

Model 3: Naive Bayes

```
## [1] 0.6604
```

```
## [1] 0.4272
```

Naive Bayes has a low training & testing accuracy

Model 4: K-NearestNeighbors

```
## [1] 1
```

```
## [1] 0.4424
```

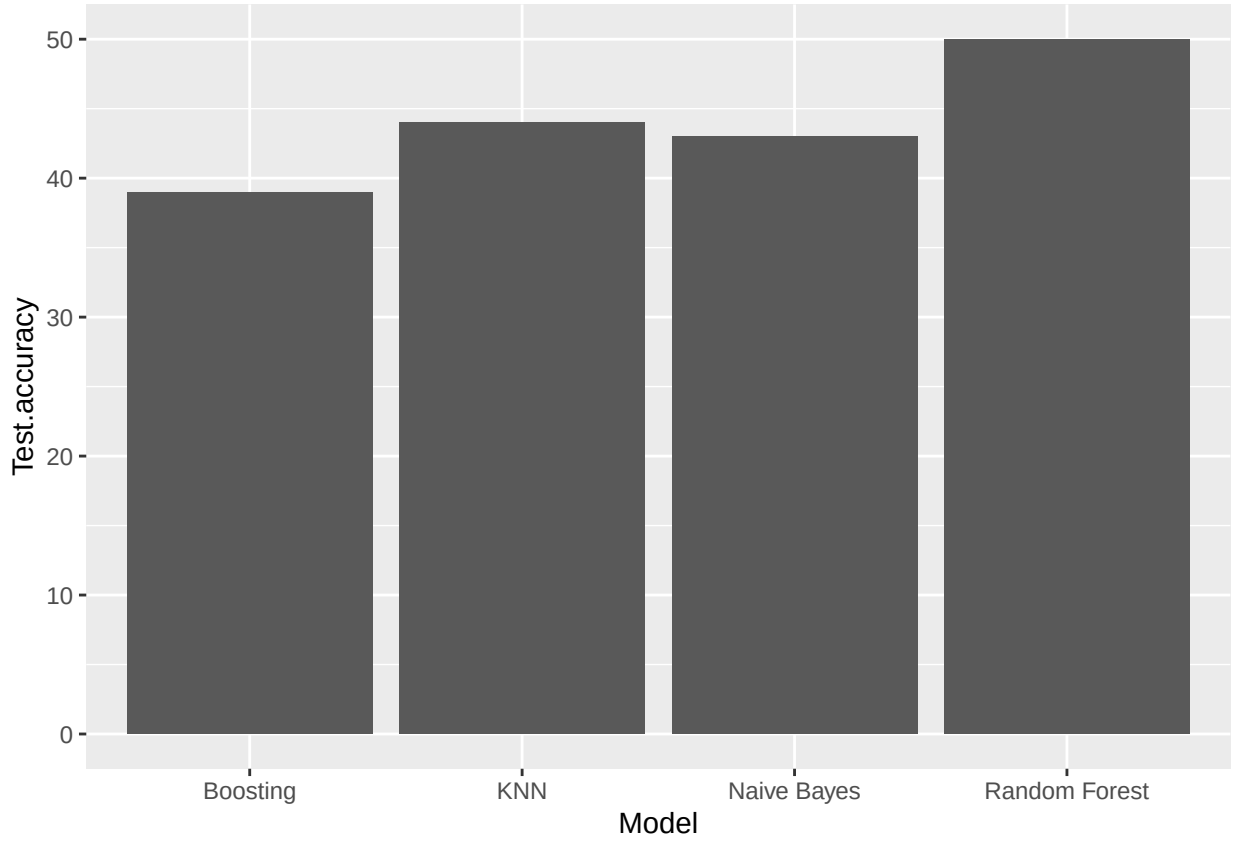
Model Comparison and Conclusion

4 different classification techniques were used to predict the author for the documents. The comparison of their results are as follows:

	Boosting	Random Forest	Naive Bayes	KNN
Training Accuracy	85%	100%	66%	100%
Test Accuracy	39%	50%	42%	43%
Computational Time	Long	Medium	Short	Medium

-Random forest provides the best accuracy out of the four methods, with a 50%. The other two methods provide lower accuracy ranging between 38-43%. -Multi-nomial logistic regression and other tree based methods can also be used for the attribution. But we have chosen 4 for this exercise.

```
##           Model Test.accuracy
## 1      Boosting            39
## 2 Random Forest            50
## 3   Naive Bayes            43
## 4           KNN            44
```



Problem 6: Association Rules Mining

```
##
## Attaching package: 'arules'
```

```
## The following object is masked from 'package:tm':
##
##     inspect

## The following objects are masked from 'package:mosaic':
##
##     inspect, lhs, rhs

## The following object is masked from 'package:dplyr':
##
##     recode

## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

Data import and Exploration

```
## [1] 9835  169

## Formal class 'transactions' [package "arules"] with 3 slots
##   ..@ data       :Formal class 'ngCMatrix' [package "Matrix"] with 5 slots
##   .. .. ..@ i       : int [1:43367] 29 88 118 132 33 157 167 166 38 91 ...
##   .. .. ..@ p       : int [1:9836] 0 4 7 8 12 16 21 22 27 28 ...
##   .. .. ..@ Dim     : int [1:2] 169 9835
##   .. .. ..@ Dimnames:List of 2
##   .. .. .. ..$ : NULL
##   .. .. .. ..$ : NULL
##   .. .. ..@ factors : list()
##   ..@ itemInfo   :'data.frame':  169 obs. of  1 variable:
##   .. ..$ labels: chr [1:169] "abrasive cleaner" "artif. sweetener" "baby cosmetics" "baby food" ...
##   ..@ itemsetInfo:'data.frame':  0 obs. of  0 variables

## transactions in sparse format with
##  6 transactions (rows) and
##  169 items (columns)
```

```
arules::inspect(grocery[1:5])
```

```
##     items
## [1] {citrus fruit,
##      margarine,
##      ready soups,
##      semi-finished bread}
## [2] {coffee,
##      tropical fruit,
##      yogurt}
## [3] {whole milk}
## [4] {cream cheese,
##      meat spreads,
##      pip fruit,
```

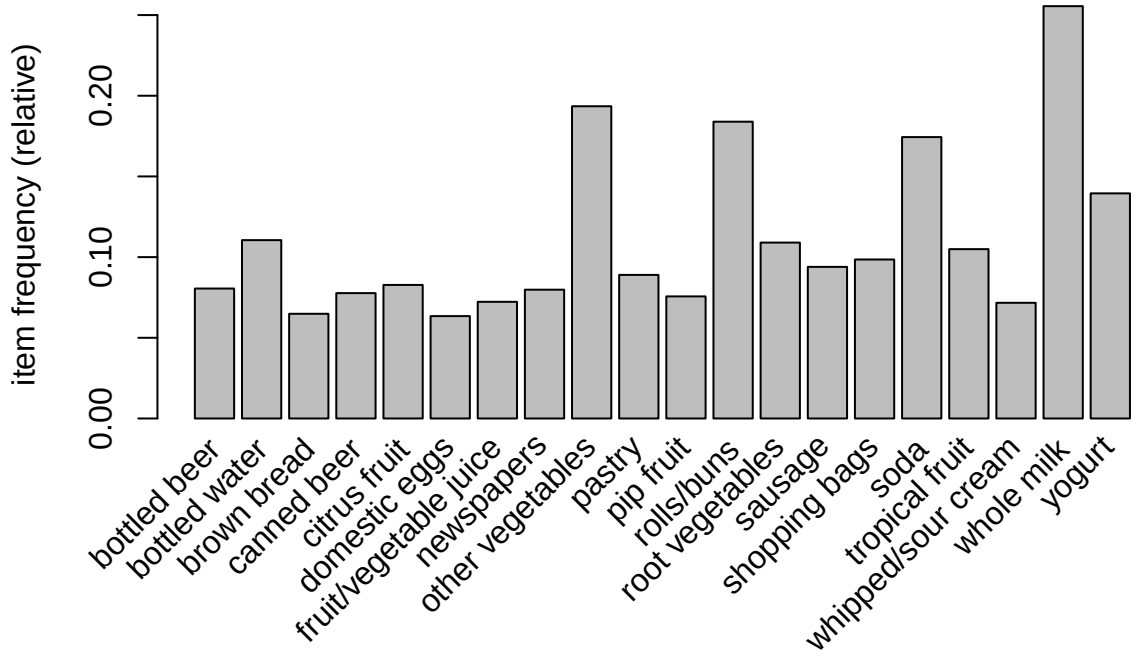
```
##       yogurt}
## [5] {condensed milk,
##      long life bakery product,
##      other vegetables,
##      whole milk}
```

```
##  [1] "abrasive cleaner" "artif. sweetener" "baby cosmetics"   "baby food"
##  [5] "bags"             "baking powder"    "bathroom cleaner" "beef"
##  [9] "berries"          "beverages"        "bottled beer"     "bottled water"
## [13] "brandy"           "brown bread"      "butter"           "butter milk"
## [17] "cake bar"         "candles"          "candy"            "canned beer"
```

```
## [1] 167
```

```
## [1] "whole milk"
```

```
## abrasive cleaner artif. sweetener   baby cosmetics        baby food
##     0.0035587189     0.0032536858     0.0006100661     0.0001016777
##             bags    baking powder bathroom cleaner             beef
##     0.0004067107     0.0176919166     0.0027452974     0.0524656838
##          berries        beverages
##     0.0332486019     0.0260294865
```



Association Rule Mining (Apriori)

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.1    0.1    1 none FALSE            TRUE       5   0.005      2
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 49
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
## sorting and recoding items ... [120 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 done [0.01s].
## writing ... [1574 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

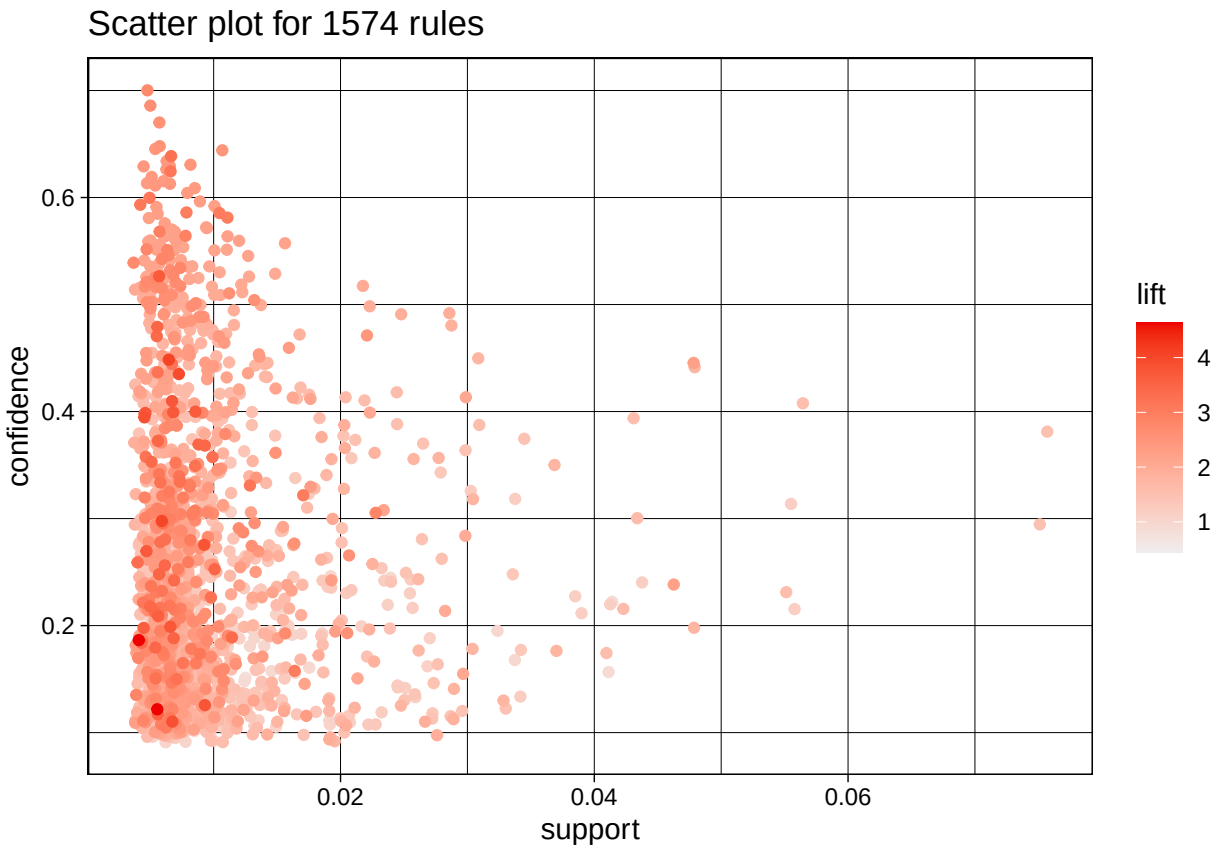
```
##      lhs              rhs                support     confidence coverage
## [1]  {cake bar}    => {whole milk}       0.005592272 0.4230769  0.01321810
## [2]  {dishes}      => {other vegetables} 0.005998983 0.3410405  0.01759024
## [3]  {dishes}      => {whole milk}       0.005287239 0.3005780  0.01759024
## [4]  {mustard}     => {whole milk}       0.005185562 0.4322034  0.01199797
## [5]  {pot plants}  => {whole milk}       0.006914082 0.4000000  0.01728521
## [6]  {chewing gum} => {soda}             0.005388917 0.2560386  0.02104728
## [7]  {chewing gum} => {whole milk}       0.005083884 0.2415459  0.02104728
## [8]  {canned fish} => {other vegetables} 0.005083884 0.3378378  0.01504830
## [9]  {pasta}       => {whole milk}       0.006100661 0.4054054  0.01504830
## [10] {herbs}       => {root vegetables}  0.007015760 0.4312500  0.01626843
##      lift      count
## [1]  1.6557746 55
## [2]  1.7625502 59
## [3]  1.1763569 52
## [4]  1.6914924 51
## [5]  1.5654596 68
## [6]  1.4683033 53
## [7]  0.9453259 50
## [8]  1.7459985 50
## [9]  1.5866145 60
## [10] 3.9564774 69
```

```
##      lhs                    rhs                     support confidence   coverage     lift count
## [1]  {ham}               => {white bread}        0.005083884  0.1953125 0.02602949 4.639851    50
## [2]  {white bread}       => {ham}                0.005083884  0.1207729 0.04209456 4.639851    50
## [3]  {citrus fruit,
##       other vegetables,
##       whole milk}        => {root vegetables}    0.005795628  0.4453125 0.01301474 4.085493    57
## [4]  {butter,
```

```
##       other vegetables}   => {whipped/sour cream} 0.005795628  0.2893401 0.02003050 4.036397    57
## [5]  {herbs}              => {root vegetables}    0.007015760  0.4312500 0.01626843 3.956477    69
## [6]  {other vegetables,
##       root vegetables}    => {onions}             0.005693950  0.1201717 0.04738180 3.875044    56
## [7]  {citrus fruit,
##       pip fruit}          => {tropical fruit}     0.005592272  0.4044118 0.01382816 3.854060    55
## [8]  {berries}            => {whipped/sour cream} 0.009049314  0.2721713 0.03324860 3.796886    89
## [9]  {whipped/sour cream} => {berries}            0.009049314  0.1262411 0.07168277 3.796886    89
## [10] {other vegetables,
##       tropical fruit,
##       whole milk}         => {root vegetables}    0.007015760  0.4107143 0.01708185 3.768074    69
## [11] {whipped/sour cream,
##       whole milk}         => {butter}             0.006710727  0.2082019 0.03223183 3.757185    66
## [12] {root vegetables,
##       whole milk,
##       yogurt}             => {tropical fruit}     0.005693950  0.3916084 0.01453991 3.732043    56
## [13] {other vegetables,
##       pip fruit,
##       whole milk}         => {root vegetables}    0.005490595  0.4060150 0.01352313 3.724961    54
## [14] {citrus fruit,
##       tropical fruit}     => {pip fruit}          0.005592272  0.2806122 0.01992883 3.709437    55
## [15] {curd,
##       tropical fruit}     => {yogurt}             0.005287239  0.5148515 0.01026945 3.690645    52
## [16] {beef,
##       other vegetables}   => {root vegetables}    0.007930859  0.4020619 0.01972547 3.688692    78
## [17] {onions,
##       other vegetables}   => {root vegetables}    0.005693950  0.4000000 0.01423488 3.669776    56
## [18] {other vegetables,
##       whipped/sour cream} => {butter}             0.005795628  0.2007042 0.02887646 3.621883    57
## [19] {domestic eggs,
##       whole milk}         => {butter}             0.005998983  0.2000000 0.02999492 3.609174    59
## [20] {hygiene articles}   => {napkins}            0.006100661  0.1851852 0.03294357 3.536498    60
```

Visualizing Apriori

```
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```




```
##      lhs                     rhs                      support confidence    coverage     lift c
## [1]  {herbs}              => {root vegetables}       0.007015760  0.4312500 0.016268429 3.956477
## [2]  {ham}                => {white bread}           0.005083884  0.1953125 0.026029487 4.639851
## [3]  {white bread}        => {ham}                   0.005083884  0.1207729 0.042094560 4.639851
## [4]  {sliced cheese}      => {sausage}               0.007015760  0.2863071 0.024504321 3.047435
## [5]  {berries}            => {whipped/sour cream}    0.009049314  0.2721713 0.033248602 3.796886
## [6]  {whipped/sour cream} => {berries}               0.009049314  0.1262411 0.071682766 3.796886
## [7]  {hygiene articles}   => {napkins}              0.006100661  0.1851852 0.032943569 3.536498
## [8]  {napkins}            => {hygiene articles}      0.006100661  0.1165049 0.052364006 3.536498
## [9]  {waffles}            => {chocolate}             0.005795628  0.1507937 0.038434164 3.039048
## [10] {chocolate}          => {waffles}               0.005795628  0.1168033 0.049618709 3.039048
## [11] {chicken}            => {frozen vegetables}     0.006710727  0.1563981 0.042907982 3.251956
## [12] {frozen vegetables}  => {chicken}               0.006710727  0.1395349 0.048093543 3.251956
## [13] {beef}               => {root vegetables}       0.017386884  0.3313953 0.052465684 3.040367
## [14] {root vegetables}    => {beef}                  0.017386884  0.1595149 0.108998475 3.040367
## [15] {onions,
##       root vegetables}    => {other vegetables}      0.005693950  0.6021505 0.009456024 3.112008
## [16] {onions,
##       other vegetables}   => {root vegetables}       0.005693950  0.4000000 0.014234875 3.669776
## [17] {other vegetables,
##       root vegetables}    => {onions}                0.005693950  0.1201717 0.047381800 3.875044
## [18] {other vegetables,
##       yogurt}             => {cream cheese}          0.005287239  0.1217799 0.043416370 3.071038
## [19] {chicken,
##       whole milk}         => {root vegetables}       0.005998983  0.3410405 0.017590239 3.128855
## [20] {frozen vegetables,
```

```
##       other vegetables}      => {root vegetables}       0.006100661  0.3428571 0.017793594 3.145522
## [21] {beef,
##       other vegetables}      => {root vegetables}       0.007930859  0.4020619 0.019725470 3.688692
## [22] {other vegetables,
##       root vegetables}       => {beef}                  0.007930859  0.1673820 0.047381800 3.190313
## [23] {beef,
##       whole milk}            => {root vegetables}       0.008032537  0.3779904 0.021250635 3.467851
## [24] {root vegetables,
##       whole milk}            => {beef}                  0.008032537  0.1642412 0.048906965 3.130449
## [25] {curd,
##       whole milk}            => {whipped/sour cream}    0.005897306  0.2256809 0.026131164 3.148329
## [26] {whipped/sour cream,
##       whole milk}            => {curd}                  0.005897306  0.1829653 0.032231825 3.434091
## [27] {curd,
##       tropical fruit}        => {yogurt}                0.005287239  0.5148515 0.010269446 3.690645
## [28] {tropical fruit,
##       yogurt}                => {curd}                  0.005287239  0.1805556 0.029283172 3.388862
## [29] {whole milk,
##       yogurt}                => {curd}                  0.010066090  0.1796733 0.056024403 3.372304
## [30] {margarine,
##       whole milk}            => {domestic eggs}         0.005185562  0.2142857 0.024199288 3.377404
## [31] {butter,
##       whole milk}            => {domestic eggs}         0.005998983  0.2177122 0.027554652 3.431409
## [32] {domestic eggs,
##       whole milk}            => {butter}                0.005998983  0.2000000 0.029994916 3.609174
## [33] {butter,
##       other vegetables}      => {whipped/sour cream}    0.005795628  0.2893401 0.020030503 4.036397
## [34] {other vegetables,
##       whipped/sour cream}    => {butter}                0.005795628  0.2007042 0.028876462 3.621883
## [35] {butter,
##       whole milk}            => {whipped/sour cream}    0.006710727  0.2435424 0.027554652 3.397503
## [36] {whipped/sour cream,
##       whole milk}            => {butter}                0.006710727  0.2082019 0.032231825 3.757185
## [37] {citrus fruit,
##       whole milk}            => {butter}                0.005083884  0.1666667 0.030503305 3.007645
## [38] {butter,
##       other vegetables}      => {root vegetables}       0.006609049  0.3299492 0.020030503 3.027100
## [39] {root vegetables,
##       whole milk}            => {butter}                0.008235892  0.1683992 0.048906965 3.038910
## [40] {whole milk,
##       yogurt}                => {butter}                0.009354347  0.1669691 0.056024403 3.013104
## [41] {domestic eggs,
##       other vegetables}      => {whipped/sour cream}    0.005083884  0.2283105 0.022267412 3.185012
## [42] {domestic eggs,
##       other vegetables}      => {root vegetables}       0.007320793  0.3287671 0.022267412 3.016254
## [43] {pip fruit,
##       whipped/sour cream}    => {other vegetables}      0.005592272  0.6043956 0.009252669 3.123610
## [44] {tropical fruit,
##       whipped/sour cream}    => {yogurt}                0.006202339  0.4485294 0.013828165 3.215224
## [45] {other vegetables,
##       tropical fruit}        => {whipped/sour cream}    0.007829181  0.2181303 0.035892222 3.042995
## [46] {root vegetables,
##       yogurt}                => {whipped/sour cream}    0.006405694  0.2480315 0.025826131 3.460127
## [47] {other vegetables,
```

```
##       yogurt}               => {whipped/sour cream}    0.010167768  0.2341920 0.043416370 3.267062
## [48] {citrus fruit,
##       pip fruit}            => {tropical fruit}        0.005592272  0.4044118 0.013828165 3.854060
## [49] {pip fruit,
##       tropical fruit}       => {citrus fruit}          0.005592272  0.2736318 0.020437214 3.306105
## [50] {citrus fruit,
##       tropical fruit}       => {pip fruit}             0.005592272  0.2806122 0.019928826 3.709437
## [51] {pip fruit,
##       root vegetables}      => {tropical fruit}        0.005287239  0.3398693 0.015556685 3.238967
## [52] {root vegetables,
##       tropical fruit}       => {pip fruit}             0.005287239  0.2512077 0.021047280 3.320737
## [53] {pip fruit,
##       yogurt}               => {tropical fruit}        0.006405694  0.3559322 0.017996950 3.392048
## [54] {other vegetables,
##       pip fruit}            => {tropical fruit}        0.009456024  0.3618677 0.026131164 3.448613
## [55] {other vegetables,
##       tropical fruit}       => {pip fruit}             0.009456024  0.2634561 0.035892222 3.482649
## [56] {citrus fruit,
##       root vegetables}      => {tropical fruit}        0.005693950  0.3218391 0.017691917 3.067139
## [57] {root vegetables,
##       tropical fruit}       => {citrus fruit}          0.005693950  0.2705314 0.021047280 3.268644
## [58] {other vegetables,
##       tropical fruit}       => {citrus fruit}          0.009049314  0.2521246 0.035892222 3.046248
## [59] {citrus fruit,
##       root vegetables}      => {other vegetables}      0.010371124  0.5862069 0.017691917 3.029608
## [60] {citrus fruit,
##       other vegetables}     => {root vegetables}       0.010371124  0.3591549 0.028876462 3.295045
## [61] {rolls/buns,
##       shopping bags}        => {sausage}               0.005998983  0.3072917 0.019522115 3.270794
## [62] {root vegetables,
##       yogurt}               => {tropical fruit}        0.008134215  0.3149606 0.025826131 3.001587
## [63] {root vegetables,
##       tropical fruit}       => {other vegetables}      0.012302999  0.5845411 0.021047280 3.020999
## [64] {other vegetables,
##       tropical fruit}       => {root vegetables}       0.012302999  0.3427762 0.035892222 3.144780
## [65] {fruit/vegetable juice,
##       other vegetables,
##       whole milk}           => {yogurt}                0.005083884  0.4854369 0.010472801 3.479790
## [66] {other vegetables,
##       whole milk,
##       yogurt}               => {fruit/vegetable juice} 0.005083884  0.2283105 0.022267412 3.158135
## [67] {other vegetables,
##       whipped/sour cream,
##       whole milk}           => {root vegetables}       0.005185562  0.3541667 0.014641586 3.249281
## [68] {other vegetables,
##       root vegetables,
##       whole milk}           => {whipped/sour cream}    0.005185562  0.2236842 0.023182511 3.120474
## [69] {other vegetables,
##       whole milk,
##       yogurt}               => {whipped/sour cream}    0.005592272  0.2511416 0.022267412 3.503514
## [70] {pip fruit,
##       root vegetables,
##       whole milk}           => {other vegetables}      0.005490595  0.6136364 0.008947636 3.171368
## [71] {other vegetables,
```

```
##       pip fruit,
##       whole milk}           => {root vegetables}       0.005490595  0.4060150 0.013523132 3.724961
## [72] {other vegetables,
##       root vegetables,
##       whole milk}           => {pip fruit}             0.005490595  0.2368421 0.023182511 3.130836
## [73] {other vegetables,
##       whole milk,
##       yogurt}               => {pip fruit}             0.005083884  0.2283105 0.022267412 3.018056
## [74] {citrus fruit,
##       root vegetables,
##       whole milk}           => {other vegetables}      0.005795628  0.6333333 0.009150991 3.273165
## [75] {citrus fruit,
##       other vegetables,
##       whole milk}           => {root vegetables}       0.005795628  0.4453125 0.013014743 4.085493
## [76] {other vegetables,
##       root vegetables,
##       whole milk}           => {citrus fruit}          0.005795628  0.2500000 0.023182511 3.020577
## [77] {root vegetables,
##       tropical fruit,
##       whole milk}           => {yogurt}                0.005693950  0.4745763 0.011997966 3.401937
## [78] {tropical fruit,
##       whole milk,
##       yogurt}               => {root vegetables}       0.005693950  0.3758389 0.015149975 3.448112
## [79] {root vegetables,
##       whole milk,
##       yogurt}               => {tropical fruit}        0.005693950  0.3916084 0.014539908 3.732043
## [80] {root vegetables,
##       tropical fruit,
##       whole milk}           => {other vegetables}      0.007015760  0.5847458 0.011997966 3.022057
## [81] {other vegetables,
##       tropical fruit,
##       whole milk}           => {root vegetables}       0.007015760  0.4107143 0.017081851 3.768074
## [82] {other vegetables,
##       tropical fruit,
##       whole milk}           => {yogurt}                0.007625826  0.4464286 0.017081851 3.200164
## [83] {other vegetables,
##       whole milk,
##       yogurt}               => {tropical fruit}        0.007625826  0.3424658 0.022267412 3.263712
## [84] {other vegetables,
##       whole milk,
##       yogurt}               => {root vegetables}       0.007829181  0.3515982 0.022267412 3.225716
## [85] {other vegetables,
##       rolls/buns,
##       whole milk}           => {root vegetables}       0.006202339  0.3465909 0.017895272 3.179778

##       lhs                         rhs                    support confidence    coverage     lift co
## [1]   {baking powder}          => {whole milk}       0.009252669  0.5229885 0.017691917 2.046793
## [2]   {oil,
##        other vegetables}       => {whole milk}       0.005083884  0.5102041 0.009964413 1.996760
## [3]   {onions,
##        root vegetables}        => {other vegetables} 0.005693950  0.6021505 0.009456024 3.112008
## [4]   {onions,
##        whole milk}             => {other vegetables} 0.006609049  0.5462185 0.012099644 2.822942
## [5]   {hygiene articles,
```

```
##       other vegetables}         => {whole milk}       0.005185562  0.5425532 0.009557702 2.123363
## [6]  {other vegetables,
##       sugar}                    => {whole milk}       0.006304016  0.5849057 0.010777834 2.289115
## [7]  {long life bakery product,
##       other vegetables}         => {whole milk}       0.005693950  0.5333333 0.010676157 2.087279
## [8]  {cream cheese,
##       yogurt}                   => {whole milk}       0.006609049  0.5327869 0.012404677 2.085141
## [9]  {chicken,
##       root vegetables}          => {other vegetables} 0.005693950  0.5233645 0.010879512 2.704829
## [10] {chicken,
##       root vegetables}          => {whole milk}       0.005998983  0.5514019 0.010879512 2.157993
## [11] {chicken,
##       rolls/buns}               => {whole milk}       0.005287239  0.5473684 0.009659380 2.142208
## [12] {coffee,
##       yogurt}                   => {whole milk}       0.005083884  0.5208333 0.009761057 2.038359
## [13] {frozen vegetables,
##       root vegetables}          => {other vegetables} 0.006100661  0.5263158 0.011591256 2.720082
## [14] {frozen vegetables,
##       root vegetables}          => {whole milk}       0.006202339  0.5350877 0.011591256 2.094146
## [15] {frozen vegetables,
##       other vegetables}         => {whole milk}       0.009659380  0.5428571 0.017793594 2.124552
## [16] {beef,
##       yogurt}                   => {whole milk}       0.006100661  0.5217391 0.011692933 2.041904
## [17] {curd,
##       whipped/sour cream}       => {whole milk}       0.005897306  0.5631068 0.010472801 2.203802
## [18] {curd,
##       tropical fruit}           => {yogurt}           0.005287239  0.5148515 0.010269446 3.690645
## [19] {curd,
##       tropical fruit}           => {other vegetables} 0.005287239  0.5148515 0.010269446 2.660833
## [20] {curd,
##       tropical fruit}           => {whole milk}       0.006507372  0.6336634 0.010269446 2.479936
## [21] {curd,
##       root vegetables}          => {other vegetables} 0.005490595  0.5046729 0.010879512 2.608228
## [22] {curd,
##       root vegetables}          => {whole milk}       0.006202339  0.5700935 0.010879512 2.231146
## [23] {curd,
##       yogurt}                   => {whole milk}       0.010066090  0.5823529 0.017285206 2.279125
## [24] {curd,
##       rolls/buns}               => {whole milk}       0.005897306  0.5858586 0.010066090 2.292845
## [25] {curd,
##       other vegetables}         => {whole milk}       0.009862735  0.5739645 0.017183528 2.246296
## [26] {pork,
##       root vegetables}          => {other vegetables} 0.007015760  0.5149254 0.013624809 2.661214
## [27] {pork,
##       rolls/buns}               => {whole milk}       0.006202339  0.5495495 0.011286223 2.150744
## [28] {frankfurter,
##       tropical fruit}           => {whole milk}       0.005185562  0.5483871 0.009456024 2.146195
## [29] {frankfurter,
##       yogurt}                   => {whole milk}       0.006202339  0.5545455 0.011184545 2.170296
## [30] {bottled beer,
##       yogurt}                   => {whole milk}       0.005185562  0.5604396 0.009252669 2.193364
## [31] {brown bread,
##       tropical fruit}           => {whole milk}       0.005693950  0.5333333 0.010676157 2.087279
## [32] {brown bread,
```

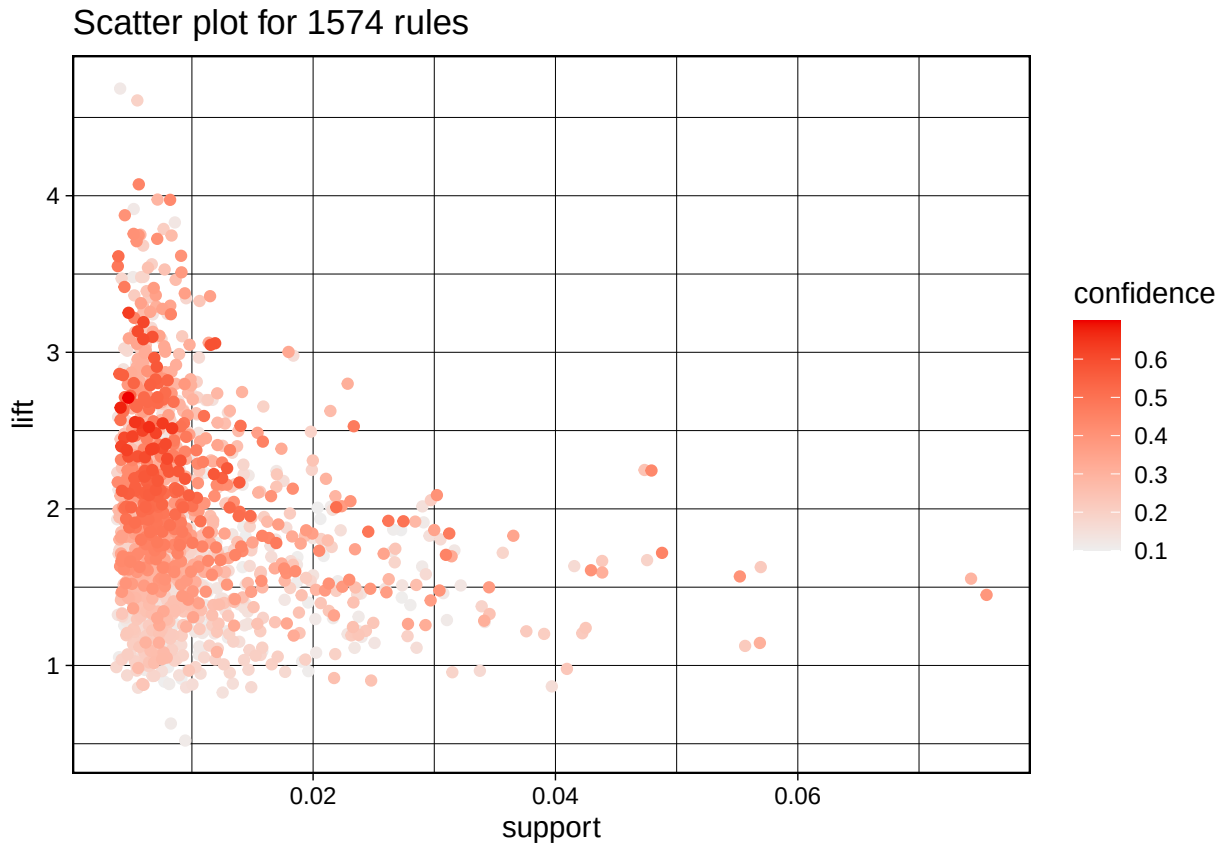
```
##        root vegetables}         => {whole milk}       0.005693950  0.5600000 0.010167768 2.191643
## [33]  {domestic eggs,
##        margarine}               => {whole milk}       0.005185562  0.6219512 0.008337570 2.434099
## [34]  {margarine,
##        root vegetables}         => {other vegetables} 0.005897306  0.5321101 0.011082867 2.750028
## [35]  {margarine,
##        rolls/buns}              => {whole milk}       0.007930859  0.5379310 0.014743264 2.105273
## [36]  {butter,
##        domestic eggs}           => {whole milk}       0.005998983  0.6210526 0.009659380 2.430582
## [37]  {butter,
##        whipped/sour cream}      => {other vegetables} 0.005795628  0.5700000 0.010167768 2.945849
## [38]  {butter,
##        whipped/sour cream}      => {whole milk}       0.006710727  0.6600000 0.010167768 2.583008
## [39]  {butter,
##        citrus fruit}            => {whole milk}       0.005083884  0.5555556 0.009150991 2.174249
## [40]  {bottled water,
##        butter}                  => {whole milk}       0.005388917  0.6022727 0.008947636 2.357084
## [41]  {butter,
##        tropical fruit}          => {other vegetables} 0.005490595  0.5510204 0.009964413 2.847759
## [42]  {butter,
##        tropical fruit}          => {whole milk}       0.006202339  0.6224490 0.009964413 2.436047
## [43]  {butter,
##        root vegetables}         => {other vegetables} 0.006609049  0.5118110 0.012913066 2.645119
## [44]  {butter,
##        root vegetables}         => {whole milk}       0.008235892  0.6377953 0.012913066 2.496107
## [45]  {butter,
##        yogurt}                  => {whole milk}       0.009354347  0.6388889 0.014641586 2.500387
## [46]  {butter,
##        other vegetables}        => {whole milk}       0.011489578  0.5736041 0.020030503 2.244885
## [47]  {newspapers,
##        root vegetables}         => {other vegetables} 0.005998983  0.5221239 0.011489578 2.698417
## [48]  {newspapers,
##        root vegetables}         => {whole milk}       0.005795628  0.5044248 0.011489578 1.974142
## [49]  {domestic eggs,
##        whipped/sour cream}      => {other vegetables} 0.005083884  0.5102041 0.009964413 2.636814
## [50]  {domestic eggs,
##        whipped/sour cream}      => {whole milk}       0.005693950  0.5714286 0.009964413 2.236371
## [51]  {domestic eggs,
##        pip fruit}               => {whole milk}       0.005388917  0.6235294 0.008642603 2.440275
## [52]  {citrus fruit,
##        domestic eggs}           => {whole milk}       0.005693950  0.5490196 0.010371124 2.148670
## [53]  {domestic eggs,
##        tropical fruit}          => {whole milk}       0.006914082  0.6071429 0.011387900 2.376144
## [54]  {domestic eggs,
##        root vegetables}         => {other vegetables} 0.007320793  0.5106383 0.014336553 2.639058
## [55]  {domestic eggs,
##        root vegetables}         => {whole milk}       0.008540925  0.5957447 0.014336553 2.331536
## [56]  {domestic eggs,
##        yogurt}                  => {whole milk}       0.007727504  0.5390071 0.014336553 2.109485
## [57]  {domestic eggs,
##        other vegetables}        => {whole milk}       0.012302999  0.5525114 0.022267412 2.162336
## [58]  {fruit/vegetable juice,
##        root vegetables}         => {other vegetables} 0.006609049  0.5508475 0.011997966 2.846865
## [59]  {fruit/vegetable juice,
```

```
##        root vegetables}          => {whole milk}       0.006507372  0.5423729 0.011997966 2.122657
## [60]  {fruit/vegetable juice,
##        yogurt}                   => {whole milk}       0.009456024  0.5054348 0.018708693 1.978094
## [61]  {pip fruit,
##        whipped/sour cream}       => {other vegetables} 0.005592272  0.6043956 0.009252669 3.123610
## [62]  {pip fruit,
##        whipped/sour cream}       => {whole milk}       0.005998983  0.6483516 0.009252669 2.537421
## [63]  {citrus fruit,
##        whipped/sour cream}       => {other vegetables} 0.005693950  0.5233645 0.010879512 2.704829
## [64]  {citrus fruit,
##        whipped/sour cream}       => {whole milk}       0.006304016  0.5794393 0.010879512 2.267722
## [65]  {sausage,
##        whipped/sour cream}       => {whole milk}       0.005083884  0.5617978 0.009049314 2.198679
## [66]  {tropical fruit,
##        whipped/sour cream}       => {other vegetables} 0.007829181  0.5661765 0.013828165 2.926088
## [67]  {tropical fruit,
##        whipped/sour cream}       => {whole milk}       0.007930859  0.5735294 0.013828165 2.244593
## [68]  {root vegetables,
##        whipped/sour cream}       => {whole milk}       0.009456024  0.5535714 0.017081851 2.166484
## [69]  {whipped/sour cream,
##        yogurt}                   => {whole milk}       0.010879512  0.5245098 0.020742247 2.052747
## [70]  {rolls/buns,
##        whipped/sour cream}       => {whole milk}       0.007829181  0.5347222 0.014641586 2.092715
## [71]  {other vegetables,
##        whipped/sour cream}       => {whole milk}       0.014641586  0.5070423 0.028876462 1.984385
## [72]  {pip fruit,
##        sausage}                  => {whole milk}       0.005592272  0.5188679 0.010777834 2.030667
## [73]  {pip fruit,
##        root vegetables}          => {other vegetables} 0.008134215  0.5228758 0.015556685 2.702304
## [74]  {pip fruit,
##        root vegetables}          => {whole milk}       0.008947636  0.5751634 0.015556685 2.250988
## [75]  {pip fruit,
##        yogurt}                   => {whole milk}       0.009557702  0.5310734 0.017996950 2.078435
## [76]  {other vegetables,
##        pip fruit}                => {whole milk}       0.013523132  0.5175097 0.026131164 2.025351
## [77]  {pastry,
##        tropical fruit}           => {whole milk}       0.006710727  0.5076923 0.013218099 1.986930
## [78]  {pastry,
##        root vegetables}          => {other vegetables} 0.005897306  0.5370370 0.010981190 2.775491
## [79]  {pastry,
##        root vegetables}          => {whole milk}       0.005693950  0.5185185 0.010981190 2.029299
## [80]  {pastry,
##        yogurt}                   => {whole milk}       0.009150991  0.5172414 0.017691917 2.024301
## [81]  {citrus fruit,
##        root vegetables}          => {other vegetables} 0.010371124  0.5862069 0.017691917 3.029608
## [82]  {citrus fruit,
##        root vegetables}          => {whole milk}       0.009150991  0.5172414 0.017691917 2.024301
## [83]  {root vegetables,
##        shopping bags}            => {other vegetables} 0.006609049  0.5158730 0.012811388 2.666112
## [84]  {sausage,
##        tropical fruit}           => {whole milk}       0.007219115  0.5182482 0.013929842 2.028241
## [85]  {root vegetables,
##        sausage}                  => {whole milk}       0.007727504  0.5170068 0.014946619 2.023383
## [86]  {root vegetables,
```

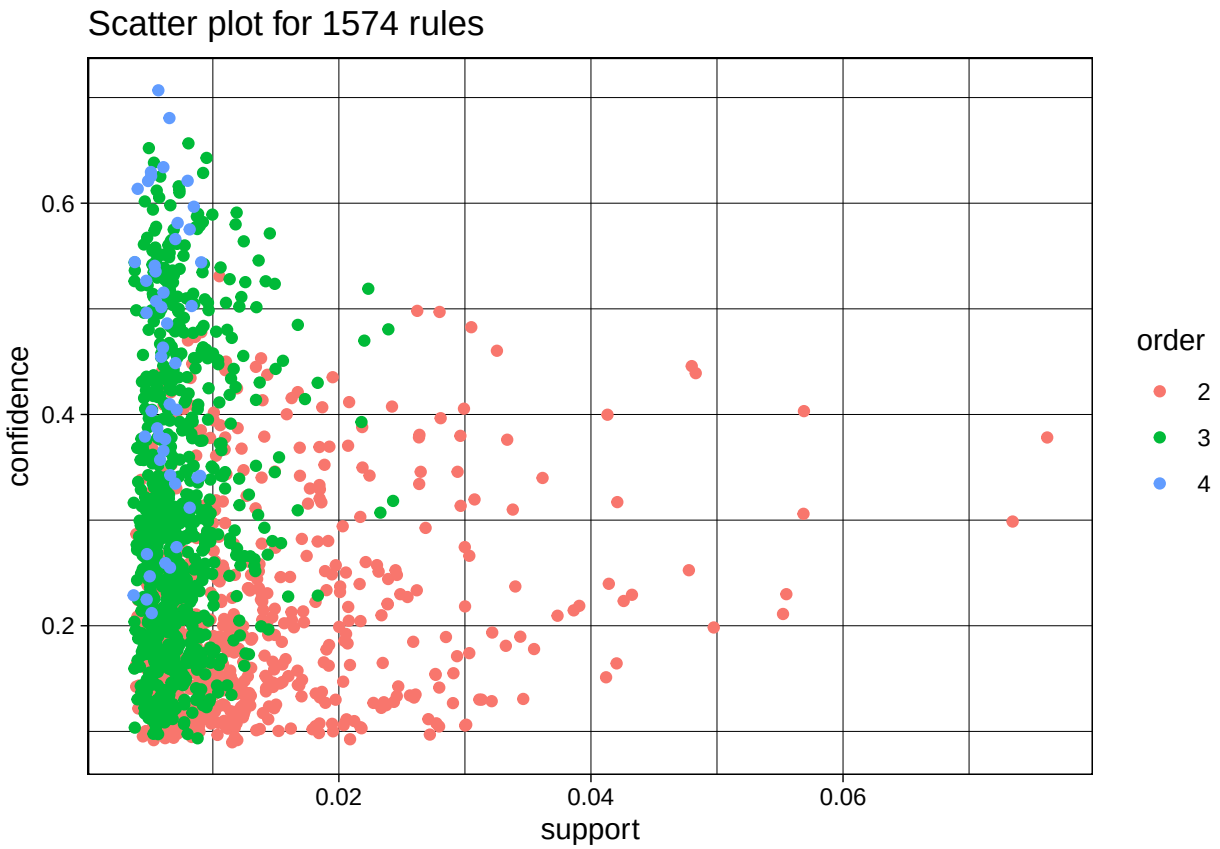
```
##        tropical fruit}          => {other vegetables} 0.012302999  0.5845411 0.021047280 3.020999
## [87]  {root vegetables,
##        tropical fruit}          => {whole milk}       0.011997966  0.5700483 0.021047280 2.230969
## [88]  {tropical fruit,
##        yogurt}                  => {whole milk}       0.015149975  0.5173611 0.029283172 2.024770
## [89]  {root vegetables,
##        yogurt}                  => {whole milk}       0.014539908  0.5629921 0.025826131 2.203354
## [90]  {rolls/buns,
##        root vegetables}         => {other vegetables} 0.012201322  0.5020921 0.024300966 2.594890
## [91]  {rolls/buns,
##        root vegetables}         => {whole milk}       0.012709710  0.5230126 0.024300966 2.046888
## [92]  {other vegetables,
##        yogurt}                  => {whole milk}       0.022267412  0.5128806 0.043416370 2.007235
## [93]  {fruit/vegetable juice,
##        other vegetables,
##        yogurt}                  => {whole milk}       0.005083884  0.6172840 0.008235892 2.415833
## [94]  {fruit/vegetable juice,
##        whole milk,
##        yogurt}                  => {other vegetables} 0.005083884  0.5376344 0.009456024 2.778578
## [95]  {other vegetables,
##        root vegetables,
##        whipped/sour cream}      => {whole milk}       0.005185562  0.6071429 0.008540925 2.376144
## [96]  {root vegetables,
##        whipped/sour cream,
##        whole milk}              => {other vegetables} 0.005185562  0.5483871 0.009456024 2.834150
## [97]  {other vegetables,
##        whipped/sour cream,
##        yogurt}                  => {whole milk}       0.005592272  0.5500000 0.010167768 2.152507
## [98]  {whipped/sour cream,
##        whole milk,
##        yogurt}                  => {other vegetables} 0.005592272  0.5140187 0.010879512 2.656529
## [99]  {other vegetables,
##        pip fruit,
##        root vegetables}         => {whole milk}       0.005490595  0.6750000 0.008134215 2.641713
## [100] {pip fruit,
##        root vegetables,
##        whole milk}              => {other vegetables} 0.005490595  0.6136364 0.008947636 3.171368
## [101] {other vegetables,
##        pip fruit,
##        yogurt}                  => {whole milk}       0.005083884  0.6250000 0.008134215 2.446031
## [102] {pip fruit,
##        whole milk,
##        yogurt}                  => {other vegetables} 0.005083884  0.5319149 0.009557702 2.749019
## [103] {citrus fruit,
##        other vegetables,
##        root vegetables}         => {whole milk}       0.005795628  0.5588235 0.010371124 2.187039
## [104] {citrus fruit,
##        root vegetables,
##        whole milk}              => {other vegetables} 0.005795628  0.6333333 0.009150991 3.273165
## [105] {root vegetables,
##        tropical fruit,
##        yogurt}                  => {whole milk}       0.005693950  0.7000000 0.008134215 2.739554
## [106] {other vegetables,
##        root vegetables,
```

```
##         tropical fruit}          => {whole milk}       0.007015760  0.5702479 0.012302999 2.231750
## [107] {root vegetables,
##         tropical fruit,
##         whole milk}              => {other vegetables} 0.007015760  0.5847458 0.011997966 3.022057
## [108] {other vegetables,
##         tropical fruit,
##         yogurt}                  => {whole milk}       0.007625826  0.6198347 0.012302999 2.425816
## [109] {tropical fruit,
##         whole milk,
##         yogurt}                  => {other vegetables} 0.007625826  0.5033557 0.015149975 2.601421
## [110] {other vegetables,
##         root vegetables,
##         yogurt}                  => {whole milk}       0.007829181  0.6062992 0.012913066 2.372842
## [111] {root vegetables,
##         whole milk,
##         yogurt}                  => {other vegetables} 0.007829181  0.5384615 0.014539908 2.782853
## [112] {other vegetables,
##         rolls/buns,
##         root vegetables}         => {whole milk}       0.006202339  0.5083333 0.012201322 1.989438
## [113] {other vegetables,
##         rolls/buns,
##         yogurt}                  => {whole milk}       0.005998983  0.5221239 0.011489578 2.043410
```

```
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```



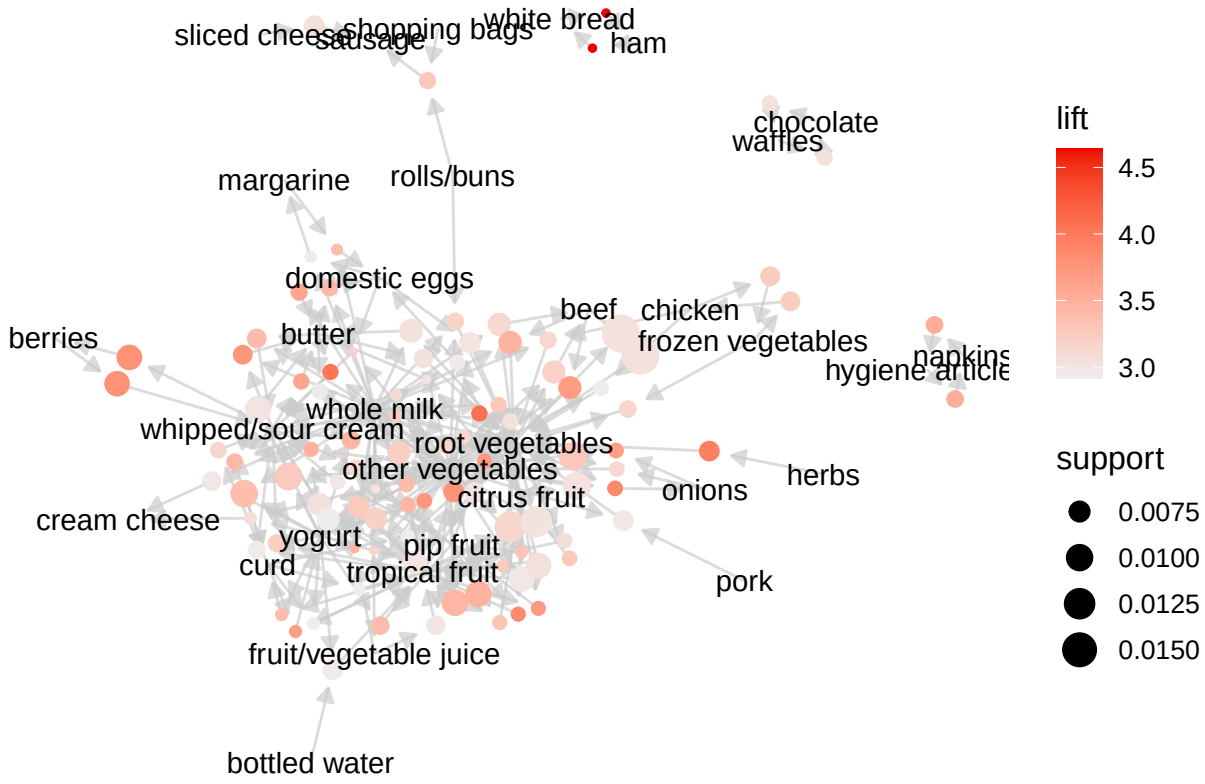

```
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```




```
##      lhs                     rhs                     support    confidence
## [1]  {whipped/sour cream} => {whole milk}          0.03223183 0.4496454
## [2]  {whole milk}         => {whipped/sour cream} 0.03223183 0.1261441
## [3]  {pip fruit}          => {whole milk}          0.03009659 0.3978495
## [4]  {whole milk}         => {pip fruit}           0.03009659 0.1177875
## [5]  {pastry}             => {whole milk}          0.03324860 0.3737143
## [6]  {whole milk}         => {pastry}              0.03324860 0.1301234
## [7]  {citrus fruit}       => {whole milk}          0.03050330 0.3685504
## [8]  {whole milk}         => {citrus fruit}        0.03050330 0.1193792
## [9]  {sausage}            => {rolls/buns}          0.03060498 0.3257576
## [10] {rolls/buns}         => {sausage}             0.03060498 0.1663903
## [11] {bottled water}      => {whole milk}          0.03436706 0.3109476
## [12] {whole milk}         => {bottled water}       0.03436706 0.1345006
## [13] {tropical fruit}     => {other vegetables}    0.03589222 0.3420543
## [14] {other vegetables}   => {tropical fruit}      0.03589222 0.1854966
## [15] {tropical fruit}     => {whole milk}          0.04229792 0.4031008
## [16] {whole milk}         => {tropical fruit}      0.04229792 0.1655392
## [17] {root vegetables}    => {other vegetables}    0.04738180 0.4347015
## [18] {other vegetables}   => {root vegetables}     0.04738180 0.2448765
## [19] {root vegetables}    => {whole milk}          0.04890696 0.4486940
## [20] {whole milk}         => {root vegetables}     0.04890696 0.1914047
## [21] {soda}               => {rolls/buns}          0.03833249 0.2198251
## [22] {rolls/buns}         => {soda}                0.03833249 0.2084024
## [23] {soda}               => {other vegetables}    0.03274021 0.1877551
## [24] {other vegetables}   => {soda}                0.03274021 0.1692065
## [25] {soda}               => {whole milk}          0.04006101 0.2297376
```

```
## [26] {whole milk}         => {soda}               0.04006101 0.1567847
## [27] {yogurt}             => {rolls/buns}         0.03436706 0.2463557
## [28] {rolls/buns}         => {yogurt}             0.03436706 0.1868436
## [29] {yogurt}             => {other vegetables}   0.04341637 0.3112245
## [30] {other vegetables}   => {yogurt}             0.04341637 0.2243826
## [31] {yogurt}             => {whole milk}         0.05602440 0.4016035
## [32] {whole milk}         => {yogurt}             0.05602440 0.2192598
## [33] {rolls/buns}         => {other vegetables}   0.04260295 0.2316197
## [34] {other vegetables}   => {rolls/buns}         0.04260295 0.2201787
## [35] {rolls/buns}         => {whole milk}         0.05663447 0.3079049
## [36] {whole milk}         => {rolls/buns}         0.05663447 0.2216474
## [37] {other vegetables}   => {whole milk}         0.07483477 0.3867578
## [38] {whole milk}         => {other vegetables}   0.07483477 0.2928770
##      coverage   lift      count
## [1]  0.07168277 1.7597542 317
## [2]  0.25551601 1.7597542 317
## [3]  0.07564820 1.5570432 296
## [4]  0.25551601 1.5570432 296
## [5]  0.08896797 1.4625865 327
## [6]  0.25551601 1.4625865 327
## [7]  0.08276563 1.4423768 300
## [8]  0.25551601 1.4423768 300
## [9]  0.09395018 1.7710480 301
## [10] 0.18393493 1.7710480 301
## [11] 0.11052364 1.2169396 338
## [12] 0.25551601 1.2169396 338
## [13] 0.10493137 1.7677896 353
## [14] 0.19349263 1.7677896 353
## [15] 0.10493137 1.5775950 416
## [16] 0.25551601 1.5775950 416
## [17] 0.10899847 2.2466049 466
## [18] 0.19349263 2.2466049 466
## [19] 0.10899847 1.7560310 481
## [20] 0.25551601 1.7560310 481
## [21] 0.17437722 1.1951242 377
## [22] 0.18393493 1.1951242 377
## [23] 0.17437722 0.9703476 322
## [24] 0.19349263 0.9703476 322
## [25] 0.17437722 0.8991124 394
## [26] 0.25551601 0.8991124 394
## [27] 0.13950178 1.3393633 338
## [28] 0.18393493 1.3393633 338
## [29] 0.13950178 1.6084566 427
## [30] 0.19349263 1.6084566 427
## [31] 0.13950178 1.5717351 551
## [32] 0.25551601 1.5717351 551
## [33] 0.18393493 1.1970465 419
## [34] 0.19349263 1.1970465 419
## [35] 0.18393493 1.2050318 557
## [36] 0.25551601 1.2050318 557
## [37] 0.19349263 1.5136341 736
## [38] 0.25551601 1.5136341 736

## Warning: Too many rules supplied. Only plotting the best 100 rules using lift
```

```
## (change control parameter max if needed)
```




```
##     lhs      rhs                support     confidence coverage   lift
## [1] {ham} => {white bread}      0.005083884 0.1953125  0.02602949 4.639851
## [2] {ham} => {tropical fruit}   0.005388917 0.2070312  0.02602949 1.973016
## [3] {ham} => {yogurt}           0.006710727 0.2578125  0.02602949 1.848095
## [4] {ham} => {rolls/buns}       0.006914082 0.2656250  0.02602949 1.444125
## [5] {ham} => {other vegetables} 0.009150991 0.3515625  0.02602949 1.816930
## [6] {ham} => {whole milk}       0.011489578 0.4414062  0.02602949 1.727509
##     count
## [1]  50
## [2]  53
## [3]  66
## [4]  68
## [5]  90
## [6] 113
```

```
##      lhs                    rhs              support confidence    coverage     lift count
## [1]  {root vegetables,
##       tropical fruit,
##       yogurt}             => {whole milk} 0.005693950  0.7000000 0.008134215 2.739554    56
## [2]  {other vegetables,
##       pip fruit,
##       root vegetables}    => {whole milk} 0.005490595  0.6750000 0.008134215 2.641713    54
## [3]  {butter,
```

```
##       whipped/sour cream} => {whole milk} 0.006710727  0.6600000 0.010167768 2.583008    66
## [4]  {pip fruit,
##       whipped/sour cream} => {whole milk} 0.005998983  0.6483516 0.009252669 2.537421    59
## [5]  {butter,
##       yogurt}             => {whole milk} 0.009354347  0.6388889 0.014641586 2.500387    92
## [6]  {butter,
##       root vegetables}    => {whole milk} 0.008235892  0.6377953 0.012913066 2.496107    81
## [7]  {curd,
##       tropical fruit}     => {whole milk} 0.006507372  0.6336634 0.010269446 2.479936    64
## [8]  {other vegetables,
##       pip fruit,
##       yogurt}             => {whole milk} 0.005083884  0.6250000 0.008134215 2.446031    50
## [9]  {domestic eggs,
##       pip fruit}          => {whole milk} 0.005388917  0.6235294 0.008642603 2.440275    53
## [10] {butter,
##       tropical fruit}     => {whole milk} 0.006202339  0.6224490 0.009964413 2.436047    61
```

```
##     lhs                      rhs       support     confidence coverage
## [1] {whipped/sour cream} => {berries} 0.009049314 0.1262411  0.07168277
##     lift     count
## [1] 3.796886 89
```

Insights

- `ham` has a very high lift value with `wheat bread`. Both products can be clubbed together as single entity or promotional discounts can be given if on both the products.
- `whole milk` is a necessity and so the customer will likely purchase this by default. Hence, such products can be stored away from the main aisles. This will cause the customer to look for Milk around the supermarket and make him/her explore other new products in the process, increasing the total billed value per customer.
- `berries` is often bought with `whipped sour cream`, thus can be placed together.
- `hygine articles` and `napkins` are also bought together.
- `citrus fruit`, `tropical fruit` and `pip fruit` are also bought together.
- Weirdly `shopping bags`, `sausages` and `sliced cheese` are bought together, shopping bags can be placed nearby sausages and sliced cheese.